PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04J 3/14

A1

(11) International Publication Number: WO 99/11003

(43) International Publication Date: 4 March 1999 (04.03.99)

(21) International Application Number: PCT/US98/17817

(22) International Filing Date: 27 August 1998 (27.08.98)

(30) Priority Data:

60/057,371	29 August 1997 (29.08.97)	US
09/018,103	3 February 1998 (03.02.98)	US

(71) Applicant *(for all designated States except US)*: EXTREME NETWORKS [US/US]; 10460 Bandley Drive, Cupertino, CA 95014 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: HADDOCK, Stephen, R. [US/US]; Extreme Networks, 10460 Bandley Drive, Cupertino, CA 95014 (US). CHUEH, Justin, N. [US/US]; Extreme Networks, 10460 Bandley Drive, Cupertino, CA 95014 (US). MERCHANT, Shehzad, T. [US/US]; Extreme Networks, 10460 Bandley Drive, Cupertino, CA 95014 (US). SMITH, Andrew, H. [GB/US]; Extreme Networks, 10460 Bandley Drive, Cupertino, CA 95014 (US). YIP, Michael [US/US]; Extreme Networks, 10460 Bandley Drive, Cupertino, CA 95014 (US).

(74) Agents: CALDWELL, Gregory, D. et al.; Blakely, Sokoloff, Taylor & Zafman, 7th floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025–1026 (US).

(81) Designated States: AL, AM, AT, AT (Utility model), AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, CZ (Utility model), DE, DK, DK (Utility model), EE, EE (Utility model), ES (Utility model), FI, GB, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SK (Utility model), SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published

With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: POLICY BASED QUALITY OF SERVICE

(57) Abstract

A flexible, policy–based, mechanism for managing, monitoring and prioritizing traffic within a network and allocating bandwidth to achieve true quality of service (QoS) is provided. A method is provided for managing bandwith allocation in a network. A packet forwarding device (100) receives information indicative of a set of traffic groups, such as MAC address, or IEEE 802.1p priority indicator or a 802.1Q frame tag, if the QoS policy is based upon individual station applications; or a physical port if the QoS policy is based purely upon topology. The packet forwarding device additionally receives information defining a QoS policy (210) for the traffic groups. After a packet is received (220), a traffic group id identified (230). The packet is scheduled (260) for transmission based upon the QoS policy for the identified traffic group rather than relying on an end–to–end signalling protocol.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

POLICY BASED QUALITY OF SERVICE

BACKGROUND OF THE INVENTION

Field of the Invention

The invention relates generally to the field of computer networking devices. More particularly, the invention relates to a flexible, policy-based mechanism for managing, monitoring, and prioritizing traffic within a network and allocating bandwidth to achieve true Quality of Service (QoS).

Description of the Related Art

Network traffic today is more diverse and bandwidth-intensive than ever before. Today's intranets are expected to support interactive multimedia, full-motion video, rich graphic images and digital photography. Expectations about the quality and timely presentation of information received from networks is higher than ever. Increased network speed and bandwidth alone will not satisfy the high demands of today's intranets.

The Internet Engineering Task Force (IETF) is working on a draft standard for the Resource Reservation Protocol (RSVP), an Internet Protocol- (IP) based protocol that allows end-stations, such as desktop computers, to request and reserve resources within and across networks. Essentially, RSVP is an end-to-end protocol that defines a means of communicating the desired Quality of Service between routers. RSVP is receiver initiated. The end-station that is receiving the data stream communicates its requirements to an adjacent router and those requirements are passed back to all intervening routers between the receiving end-station and the source of the data stream and finally to the source of the data stream itself. Therefore, it should be apparent that RSVP must be implemented across the whole network. That is, both end-

stations (e.g., the source and destination of the data stream) and every router in between should be RSVP compliant in order to accommodate the receiving end-station's request.

While RSVP allows applications to obtain some degree of guaranteed performance, it is a first-come, first-served protocol, which means if there are no other controls within the network, an application using RSVP may reserve and consume resources that could be needed or more effectively utilized by some other mission-critical application. A further limitation of this approach to resource allocation is the fact that end-stations and routers must be altered to be RSVP compliant. Finally, RSVP lacks adequate policy mechanisms for allowing differentiation between various traffic flows. It should be appreciated that without a policy system in place, the network manager loses control.

Recent attempts to facilitate traffic differentiation and prioritization include draft standards specified by the Institute of Electrical and Electronics Engineers (IEEE). The IEEE 802.1Q draft standard provides a packet format for an application to specify which Virtual Local Area Network (VLAN) a packet belongs to and the priority of the packet. The IEEE 802.1p committee provides a guideline to classify traffic based on a priority indicator in an 802.1Q frame tag. This allows VLANs to be grouped into eight different traffic classes or priorities. The IEEE 802.1p committee does not, however, define the mechanism to service these traffic classes.

What is needed is a way to provide true Quality of Service ("QoS") in a network employing a non-deterministic access protocol, such as an Ethernet network, that not only has the ability to prioritize and service different traffic classes, but additionally provides bandwidth management and guarantees a quantifiable measure of service for packets associated with a particular traffic class. More specifically, with respect to bandwidth management, it is desirable to employ a weighted fair queuing delivery schedule which shares available bandwidth so that high priority traffic is usually sent first, but low priority traffic is still guaranteed an acceptable

minimum bandwidth allocation. Also, it is desirable to centralize the control over bandwidth allocation and traffic priority to allow for QoS without having to upgrade or alter end-stations and existing routers as is typically required by end-to-end protocol solutions. Further, it would be advantageous to put the control in the hands of network managers by performing bandwidth allocation and traffic prioritization based upon a set of manager-defined administrative policies. Finally, since there are many levels of control a network manager may elect to administer, it is desirable to provide a variety of scheduling mechanisms based upon a core set of QoS profile attributes.

BRIEF SUMMARY OF THE INVENTION

A flexible, policy-based, mechanism for managing, monitoring, and prioritizing traffic within a network and allocating bandwidth to achieve true Quality of Service (QoS) is described. According to one aspect of the present invention, a method is provided for managing bandwidth allocation in a network that employs a non-deterministic access protocol. A packet forwarding device receives information indicative of a set of traffic groups. The packet forwarding device additionally receives parameters, such as bandwidth and priority parameters, corresponding to the traffic groups. After receiving a packet associated with one of the traffic groups on a first port, the packet forwarding device schedules the packet for transmission from a second port based upon parameters corresponding to the traffic group with which the packet is associated. Advantageously, in this manner, a weighted fair queuing schedule that shares bandwidth according to some set of rules may be achieved.

According to another aspect of the present invention, a method is provided for managing bandwidth allocation and traffic prioritization in a packet forwarding device. The packet forwarding device receives information indicative of a set of traffic groups. The packet

forwarding device additionally receives information defining a Quality of Service (QoS) policy for the traffic groups. After a packet is received by the packet forwarding device, a traffic group with which the packet is associated is identified. Subsequently, rather than relying on an end-to-end signaling protocol for scheduling, the packet is scheduled for transmission based upon the QoS policy for the identified traffic group. Therefore, bandwidth allocation and traffic prioritization are based upon a set of administrative policies over which the network manager retains control.

According to yet another aspect of the present invention, a number of QoS queues are provided at each port of the packet forwarding device. A current bandwidth metric is determined for each of the QoS queues for a particular port. The QoS queues are divided into two groups based upon their respective bandwidth metrics and their respective minimum bandwidth requirements. Subsequently, the groups are used as a first level arbitration mechanism to select a QoS queue that will source the next packet.

Other features of the present invention will be apparent from the accompanying drawings and from the detailed description which follows.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

Figure 1A is a simplified block diagram of an exemplary switch architecture in which one embodiment of the present invention may be implemented.

Figure 1B is a logical view of the interaction between switch processing blocks according to one embodiment of the present invention.

Figure 2 is a flow diagram illustrating high level bandwidth management and traffic prioritization processing according to one embodiment of the present invention.
according to one embodiment of the present invention.

Figure 3 is a flow diagram illustrating periodic evaluation of QoS categories according to one embodiment of the present invention.

Figure 4 is a flow diagram illustrating next packet scheduling according to one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

A flexible, policy-based, mechanism for managing, monitoring, and prioritizing traffic within a network and allocating bandwidth to achieve true Quality of Service (QoS) is described. "Quality of Service" in this context essentially means that there is a quantifiable measure of the service being provided. The measure of service being provided may be in terms of a packet loss rate, a maximum delay, a committed minimum bandwidth, or a limited maximum bandwidth, for example.

In the present invention, a number of QoS queues may be provided at each port of a packet forwarding device, such as a Local Area Network (LAN) switch. Based upon a set of QoS parameters, various types of traffic can be distinguished and associated with particular QoS queues. For example, packets associated with a first traffic group may be placed onto a first QoS queue and packets associated with another traffic group may be placed onto a second QoS queue. When a port is ready to transmit the next packet, a scheduling mechanism may be employed to

select which QoS queue of the QoS queues associated with the port will provide the next packet for transmission.

In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without some of these specific details. In other instances, well-known structures and devices are shown in block diagram form.

The present invention includes various steps, which will be described below. The steps of the present invention may be performed by hardware components or may be embodied in machine-executable instructions, which may be used to cause a general-purpose or special-purpose processor programmed with the instructions to perform the steps. Alternatively, the steps may be performed by a combination of hardware and software. While, embodiments of the present invention will be described with reference to a high speed Ethernet switch, the method and apparatus described herein are equally applicable to other types of network devices or packet forwarding devices.

An Exemplary Switch Architecture

An overview of the architecture of a switch 100 in which one embodiment of the present invention may be implemented is illustrated by Figure 1A. The central memory architecture depicted includes multiple ports 105 and 110 each coupled via a channel to a filtering/forwarding engine 115. Also coupled to the filtering/forwarding engine 115 is a forwarding database 120, a packet Random Access Memory (RAM) 125, and a Central Processing Unit (CPU) 130.

According to one embodiment, each channel is capable of supporting a data transfer rate of one gigabit per second in the transmit direction and one gigabit per second in the receive direction, thereby providing 2 Gb/s full-duplex capability per channel. Additionally, the

channels may be configured to support one Gigabit Ethernet network connection or eight Fast Ethernet network connections.

The filtering/forwarding engine 115 includes an address filter (not shown), a switch matrix (not shown), and a buffer manager (not shown). The address filter may provide bridging, routing, Virtual Local Area Network (VLAN) tagging functions, and traffic classification. The switch matrix connects each channel to a central memory such as packet RAM 125. The buffer manager controls data buffers and packet queue structures and controls and coordinates accesses to and from the packet RAM 125.

The forwarding database 120 may store information useful for making forwarding decisions, such as layer 2 (e.g., Media Access Control (MAC) layer), layer 3 (e.g., Network layer), and/or layer 4 (e.g., Transport layer) forwarding information, among other things. The switch 100 forwards a packet received at an input port to an output port by performing a search on the forwarding database using address information contained within the header of the received packet. If a matching entry is found, a forwarding decision is constructed that indicates to which output port the received packet should be forwarded, if any. Otherwise, the packet is forwarded to the CPU 130 for assistance in constructing a forwarding decision.

The packet RAM 125 provides buffering for packets and acts as an elasticity buffer for adapting between incoming and outgoing bandwidth differences. Packet buffering is discussed further below.

Logical View of Exemplary Switch Processing

Figure 1B is a logical view of the interaction between exemplary switch processing blocks that may be distributed throughout the switch 100. For example, some of the processing may be performed by functional units within the ports of the switch and other processing may be performed by the CPU 130 or by the address filter/switch matrix/buffer manager 115. In any

event, the processing can be conceptually divided into a first group of functions 160 dedicated to input processing and a second group of functions 185 dedicated to output processing. According to the present embodiment, the first group 160 includes a comparison engine 155, an enqueue block 161, a packet classification block 150, and a buffer manager 165. The second group 185 includes a dequeue block 162, a Quality of Service (QoS) category evaluation block 175, and a scheduler 170.

Additionally, a user interface (UI) 145 may be provided for receiving various parameters from the network manager. The UI may be text based or graphical. In one embodiment, the UI 145 may include an in-band HyperText Markup Language (HTML) browser-based management tool which may be accessed by any standard web browser. In any event, the goal of the UI 145 is to separate high-level policy components, such as traffic grouping and QoS profiles from the details of the internal switch hardware. Thus, user configuration time is minimized and a consistent interface is provided to the user.

The UI 145 receives information indicative of one or more traffic groups. This information may be provided by the network manager. There are several ways to define a traffic group. Table 1 below illustrates a variety of traffic classification schemes that may be supported by the UI 145.

Traffic Classification

Policy Based Upon	Traffic Group Definition	OSI Layer
Applications	TCP Session UDP Session RSVP Flow	Transport Layer
Network Layer Topology or Groups of Users	Network Layer Protocol Subnet or IP Address VLAN Identifier	Network Layer
End-Station Applications	MAC Address 802.1p or 802.1Q	Link Layer
Physical Topology	Physical Port	Physical Layer

Table 1

The information used to identify a traffic group typically depends upon what terms the QoS policy is defined. If the QoS policy is based on applications, traffic groups may be differentiated at the Transport layer by Transmission Control Protocol (TCP) session or User Datagram Protocol (UDP) session. For example, the network manager may provide information indicative of TCP source and destination ports and IP source and destination addresses to identify traffic groups. However, if the QoS policy is based upon the Network layer topology or groups of users, traffic group definition may be more convenient by supplying information regarding the Network layer protocol, such as Internet Protocol (IP) or Internetwork Packet Exchange (IPX), the subnet or IP addresses, or VLAN identifiers. If the QoS policy is defined by end-station applications, then Media Access Control (MAC) addresses, IEEE 802.1p priority indications, or IEEE 802.1Q frames may be employed to identify traffic groups. Finally, if the QoS policy is physical topology based, physical port identifiers may be used to differentiate traffic groups.

It should be noted that Table 1 merely presents an exemplary set of traffic group identification mechanisms. From the examples presented herein, additional, alternative, and equivalent traffic grouping schemes and policy considerations will be apparent to those of ordinary skill in the art. For example, other state information may be useful for purposes of packet classification, such as the history of previous packets, the previous traffic load, the time of day, etc.

It is appreciated that traffic classifications based upon the traffic group definitions listed above may result in overlap. Should the network manager define overlapping traffic groups, the UI 145 may issue an error message and reject the most recent traffic group definition, the UI 145 may issue a warning message to the network manager and allow the more specific traffic group

definition to override a conflicting general traffic group definition, or the UI 145 may be configured to respond in another manner.

A number of QoS queues 180 may be provided at each of the ports of a packet forwarding device. In one embodiment, a mapping of traffic groups to QoS queues 180 may be maintained. As traffic groups are provided by the network manager, the UI 145 updates the local mapping of traffic groups to QoS queues 180. This mapping process may be a one-to-one mapping of the traffic groups defined by the network manager to the QoS queues 180 or the mapping process may be more involved. For example, there may be more traffic groups than QoS queues 180, in which case, more than one traffic group will be mapped to a single QoS queue. Some consolidation rules for combining multiple traffic groups into a single QoS queue will be discussed below.

At any rate, by providing a layer of abstraction in this manner, the network manager need not be burdened with the underlying implementation details, such as the number of QoS queues per port and other queuing parameters. Another advantage achieved by this layer of abstraction between the traffic group definitions and the physical QoS queues is the fact that the UI 145 is now decoupled from the underlying implementation. Therefore, the UI 145 need not be updated if the hardware QoS implementation changes. For example, software providing for traffic group definition need not be changed simply because the number of QoS queues per port provided by the hardware changes.

The input data stream is received by the comparison engine 155 from input switch ports (not shown). Under the direction of the packet classification process 150, the comparison engine 155 determines with which of the previously defined traffic groups a packet in the data stream is associated. The packet classification block 150 may employ the traffic group indications provided by the network manager to provide the comparison engine 155 with information regarding locations and fields to be compared or ignored within the header of a received packet,

for example. It should be appreciated if the comparison required for traffic classification is straightforward, such as in a conventional packet forwarding device, then the comparison engine 155 and the packet classification block 150 may be combined.

The packet classification block 150 in conjunction with the UI 145 provide a network manager with a flexible mechanism to control traffic prioritization and bandwidth allocation through the switch 100. Importantly, no end-to-end signaling protocol needs to be implemented by the network devices. For example, the end-station that is to receive the data stream need not reserve bandwidth on each of the intermediate devices between it and the source of the data stream. Rather, a packet forwarding device employing the present invention can provide some benefit to the network without requiring routers and/or end-stations to do anything in particular to identify traffic. Thus, traffic priority may be enforced by the switch 100 and QoS may be delivered to applications without altering routers or end-stations.

According to one embodiment, the buffer manager 165 participates in policy based QoS by controlling the allocation of buffers within the packet RAM 125. Buffers may be dynamically allocated to QoS queues 180 as needed, within constraints established by QoS profile attributes, which are discussed below. The buffer manager 165 may maintain several programmable variables for each QoS queue. For example, a Minimum Buffer Allocation and a Maximum Queue Depth may be provided for each QoS queue. The Minimum Buffer Allocation essentially reserves some minimum number of buffers in the packet RAM 125 for the QoS queue with which it is associated. The Maximum Queue Depth establishes the maximum number of buffers that can be placed on a given QoS queue. The buffer manager 165 also maintains a Current Queue Depth for each QoS queue to assure the maximum depth is not exceeded. For example, before allowing a buffer to be added to a given QoS queue, the buffer manager 165 may compare the Maximum Queue Depth to the Current Queue Depth to ensure the Maximum Queue Depth is not exceeded.

Variables are also maintained for tracking free buffers in the packet RAM 125. At initialization, a Buffers Free Count contains the total number of buffers available in the packet RAM 125 and a Buffers Reserved Count contains the sum of the minimum buffer allocations for the QoS queues 180. As packets are received they are stored in free buffers, and the Buffers Free Count is decremented by the number of buffers used for such storage. After the appropriate QoS queue has been identified the buffer manager 165 instructs the enqueue block 161 to add the packet to the QoS queue. The enqueue block 161 links the packet to the identified queue provided that the Current Queue Depth is less than the Maximum Queue Depth and either (1) the Current Queue Depth is less than the Minimum Buffer Allocation or (2) the Buffers Reserved Count is less than the Buffers Free Count. Therefore, if a QoS queue exceeds its reserve of buffers (e.g., Minimum Buffer Allocation), to the extent that additional buffers remain free, the QoS queue may continue to grow. Otherwise, the enqueue block 161 will discard the packet, the buffers are returned to the free pool, and the Buffers Free Count is increased by the number of buffers that would have been consumed by the packet. When a packet is successfully linked to a QoS queue, the Current Queue Depth for that QoS queue is increased by the number of buffers used by the packet. If, prior to the addition of the packet to the queue, the Current Queue Depth was less than the Minimum Buffer Allocation then the Buffers Reserved Count is decreased by the lesser of (1) the number of buffers in the packet or (2) the difference between the Current Queue Depth and the Minimum Buffer Allocation.

The QoS category evaluation process 175 separates the QoS queues into a plurality of categories based upon a set of bandwidth parameters. The scheduler 170 uses the grouping provided by the QoS category evaluation process 175 to select an appropriate QoS queue for sourcing the next packet for a particular port. The evaluation of QoS queue categories may be performed periodically or upon command by the scheduler 170, for example. Periodic evaluation of QoS categories and scheduling is discussed in further detail below.

Responsive to the scheduler 170 the dequeue block 162 retrieves a packet from a specified QoS queue. After the packet has been transmitted, the buffer variables are updated. The Buffers Free Count is increased and the Current Queue Depth is decreased by the number of buffers utilized to store the packet. If the resulting Current Queue Depth is less than the Minimum Buffer Allocation, then the Buffers Reserved Count is increased by the lesser of the number of buffers utilized to store the packet or the difference between the Current Queue Depth and the Minimum Buffer Allocation.

QoS Profile Attributes

Setting QoS policy is a combination of identifying traffic groups and defining QoS profiles for those traffic groups. According to one embodiment, each individual traffic group may be associated with a QoS profile. However, in alternative embodiments, multiple traffic groups may share a common QoS profile. Having described traffic group classification and identification above, QoS profile attributes (also referred to as parameters) will now be discussed.

Several queuing mechanisms may be implemented using one or more of the following parameters associated with a traffic group: (1) minimum bandwidth, (2) maximum bandwidth, (3) peak bandwidth, (4) maximum delay, and (5) relative priority. In general, the minimum, maximum, and peak bandwidth parameter may be expressed in Mbps, a percentage of total bandwidth, or any other convenient representation.

Minimum bandwidth indicates the minimum amount of bandwidth a particular traffic group needs to be provided over a defined time period. If the sum of the minimum bandwidths for all traffic groups defined is less than 100% of the available bandwidth, then the scheduling processing, discussed below, can assure that each traffic group will receive at least the minimum bandwidth requested.

Maximum bandwidth is the maximum sustained bandwidth the traffic group can realize over a defined time period. In contrast, peak bandwidth represents the bandwidth a traffic group may utilize during a particular time interval in excess of the maximum bandwidth. The peak bandwidth parameter may be used to limit traffic bursts for the traffic group with which it is associated. The peak bandwidth also determines how quickly the traffic group's current bandwidth will converge to the maximum bandwidth. By providing a peak bandwidth value that is much higher than the maximum bandwidth, if sufficient bandwidth is available, the maximum bandwidth will be achieved relatively quickly. In contrast, a peak bandwidth that is only slightly higher that the maximum bandwidth will cause the convergence to the maximum bandwidth to be more gradual.

Maximum delay specifies a time period beyond which further delay cannot be tolerated for the particular traffic group. Packets comprising the traffic group that are forwarded by the switch 100 are guaranteed not to be delayed by more than the maximum delay specified.

Relative priority defines the relative importance of a particular traffic group with respect to other traffic groups. As will be discussed further below, within the same QoS category, traffic groups with a higher priority are preferred over those with lower priorities.

This small set of parameters in combination with the variety of traffic classification schemes gives a network manager enormous control and flexibility in prioritizing and managing traffic flowing through packet forwarding devices in a network. For example, the QoS profile of a video traffic group, identified by UDP session, might be defined to have a high priority and a minimum bandwidth of 5 Mbps, while the QoS profile of an engineering traffic group, identified by VLAN, may be set to a second priority, a minimum bandwidth of 30 Mbps, a maximum bandwidth of 50 Mbps, and a peak bandwidth of 60 Mbps. Concurrently, the QoS profile of a World Wide Web (WWW) traffic group, identified by protocol (e.g., IP), may be set to have a

low priority, a minimum bandwidth of 0 Mbps, a maximum bandwidth of 100%, and a peak bandwidth of 100%.

Consolidation Rules

It was mentioned earlier that multiple traffic groups may be mapped to a single QoS queue. This may be accomplished by maintaining an independent set of variables (e.g., minimum bandwidth, maximum bandwidth, peak bandwidth, maximum delay, and relative priority) for each QoS queue in addition to those already associated with each traffic group and following the general consolidation rules outlined below.

Briefly, when the mapping from traffic groups to QoS queues is one-to-one, the determination of a particular QoS queues' attributes is straightforward. The QoS queue's attributes simply equal the traffic group's attributes. However, when combining multiple traffic groups that do not share a common QoS profile onto a single QoS queue, the following general consolidation rules are suggested: (1) add minimum attributes of the traffic groups being combined to arrive at an appropriate minimum attribute for the target QoS queue (e.g., the QoS queue in which the traffic will be merged), (2) use the largest of maximum attributes to arrive at an appropriate value for a maximum attribute for the target QoS queue, and (3) avoid merging traffic groups that have different relative priorities. This last rule suggests the number of priority levels provided should be less than or equal to the number of QoS queues supported by the implementation to assure traffic groups with different priorities are not combined in the same QoS queue.

Importantly, when a network manager has determined that multiple traffic groups will share a common QoS profile, the consolidation rules need not apply, as the network manager has already, in effect, manually consolidated the parameters.

Bandwidth Management and Traffic Prioritization

Having described an exemplary environment in which one embodiment of the present invention may be implemented, bandwidth management and traffic prioritization will now be described with reference to Figure 2. Figure 2 is a flow diagram illustrating the high level bandwidth management and traffic prioritization processing according to one embodiment of the present invention. In this embodiment, at step 210, a manager-defined QoS policy may be received via the UI 145, for example. The QoS policy is a combination of traffic groups and QoS profile attributes corresponding to those traffic groups.

At step 220, a packet is received by the switch 100. Before the packet can be placed onto a QoS queue for transmission, the traffic group to which the packet belongs is identified at step 230. Typically, information in the packet header, for example, can be compared to the traffic group criteria established by the network manager to identify the traffic group to which the packet belongs. This comparison or matching process may be achieved by programming filters in the switch 100 that allow classification of traffic. According to one embodiment, the packet may be identified using the traffic group definitions listed in Table 1.

At step 250, enqueue processing is performed. The packet is added to the rear of the appropriate QoS queue for the identified traffic group. Importantly, if a maximum delay has been assigned to the traffic group with which the packet is associated, then the packet should either be dropped or transmitted within the period specified. According to one embodiment, this may be accomplished by limiting the depth (also referred to as length) of the corresponding QoS queue. Given the minimum bandwidth of the QoS queue and the maximum delay the traffic group can withstand, a maximum depth for the QoS queue can be calculated. If the QoS queue length remains less than or equal to the maximum length, then the packet is added to the QoS queue. However, if the QoS queue length would exceed the maximum length by the addition, then the packet is dropped.

At step 260, scheduling is performed. The scheduling/dequeuing processing involves determining the appropriate QoS queue group, selecting the appropriate QoS queue within that QoS queue group, and removing the packet at the front of the selected QoS queue. This selected packet will be the next packet the port transmits. Scheduling will be discussed further below.

Evaluation of QoS Categories

According to one embodiment of the present invention, it is advantageous to divide the QoS queues into at least two categories. The categories may be defined based upon the maximum bandwidth, the minimum bandwidth, the peak bandwidth, and the "current bandwidth." The current bandwidth should not be mistaken for a bandwidth at an instant in time, rather the current bandwidth is a moving average that is updated periodically upon the expiration of a predetermined time period. Empirical data suggests this predetermined time period should be on the order of ten packet times, wherein a packet time is the time required to transmit a packet. However, depending upon the environment and the nature of the traffic, a value in the range of one to one hundred packet times may be more suitable.

The members of the first category ("Category A") are those QoS queues which have a current bandwidth that is below their peak bandwidth and below their minimum bandwidth. Members of the second category ("Category B") include those QoS queues that have a current bandwidth that is greater than or equal to their minimum bandwidth, but less than both their maximum bandwidth and their peak bandwidth. The remaining QoS queues (e.g., those having a current bandwidth that is greater than or equal to either the peak bandwidth or the maximum bandwidth) are ineligible for transmission. These QoS queues that are ineligible for transmission can be considered a third category ("Category C"). With this overview of QoS categories, an exemplary process for periodic evaluation of QoS categories will now be described.

Figure 3 is a flow diagram illustrating periodic evaluation of QoS categories according to one embodiment of the present invention. In this embodiment, at step 310, processing loops until the predetermined evaluation time period has expired. For example, a test may be performed to determine if the current time is greater than or equal to the last evaluation time plus the predetermined evaluation time interval. Alternatively, the evaluation process may be triggered by an interrupt. In any event, when it is time to evaluate the QoS queue categorization, processing continues with step 320.

It will be appreciated that the time interval chosen for the predetermined evaluation time period should not be too long or too short. If the time interval is too long, one QoS queue might be allowed to monopolize the link until its maximum bandwidth is achieved while other QoS queues remain idle. If the time interval is too short, transmitting a single packet or remaining idle for a single packet time may cause the QoS queue to become a member of a different QoS category (e.g., the single transmission may cause the current bandwidth to exceed the maximum bandwidth or the single idle time may cause the current bandwidth to fall below the minimum bandwidth) because the moving average moves very quickly over short time intervals.

At step 330, the current bandwidth for a particular QoS queue is set to the current bandwidth for that QoS queue as calculated in the previous time interval multiplied by a first weighting factor plus the actual bandwidth that particular QoS queue received in the previous time interval multiplied by a second weighting factor, wherein the weighting factors may be selected to achieve the desired level of responsiveness in the current bandwidth metric. For example, it may be desirable to have the current bandwidth converge to within a certain percentage of a sustained bandwidth if that bandwidth has been sustained for a certain amount of time. Exemplary weighting factors are in the form (w-1)/w and 1/w, respectively. Using weighting factors of 15/16 for the first weighting factor and a value of 1/16 for the second weighting factor, for example, the current bandwidth will reflect 50% of a step within 13 time

intervals, 80% of a step within 27 time intervals, and will be within 2% of the sustained bandwidth in approximately 63 time intervals (assuming a maximum and peak bandwidth of 100%). Alternative ratios and current bandwidth metrics will be apparent to those of ordinary skill in the art.

After the current bandwidth has been evaluated for a QoS queue, at step 340, the QoS queue bandwidth parameters can be compared to the current bandwidth to determine to which QoS category the QoS queue belongs. As described above, if (CURR_BW < PEAK_BW) and (CURR_BW < MIN_BW), then the QoS queue is associated with Category A at step 350. If (CURR_BW MIN_BW) and ((CURR_BW < MAX_BW) and (CURR_BW < PEAK_BW)), then the QoS queue is associated with Category B at step 360. If (CURR_BW PEAK_BW) or (CURR_BW MAX_BW), then the QoS queue is associated with Category C at step 370.

At step 380, if all of the QoS queues have been evaluated, then processing branches to step 310; otherwise, processing continues with step 330.

Scheduling Processing

Briefly, at each port, three levels of arbitration may be employed to select the appropriate QoS queue from which to transmit the next packet. The first level of arbitration selects among the QoS categories. Category A is given priority if any member QoS queues have one or more pending packets. Otherwise, a QoS queue with one or more pending packets of Category B is selected. According to one embodiment, the relative priority assigned to each QoS queue may be used as a second level of arbitration. In this manner, when multiple QoS queues satisfy the first level arbitration, a higher priority QoS queue is favored over a lower priority QoS queue. Finally, when there is a tie at the second level of arbitration (e.g., two or more QoS queues in the same QoS category have the same relative priority), a round robin or least recently used (LRU)

scheme may be employed to select from among the two or more QoS queues until the QoS categories are evaluated.

Assuming a periodic evaluation of QoS categories is being performed, the scheduling processing need not include such evaluation and the scheduling processing may be performed as illustrated by Figure 4, according to one embodiment of the present invention. In the embodiment depicted, at step 410, processing loops until the port associated with the group of QoS queues being evaluated indicates it is ready to receive the next packet for transmission. For example, the port may be polled to determine its transmission status. Alternatively, the scheduling process may be triggered by an interrupt. In any event, when the port is ready for the next packet, processing continues with step 420.

At step 420, a QoS category is selected from which a QoS queue will provide the next packet for transmission. As described above, priority is given to the category containing QoS queues with pending data that are below the peak bandwidth and minimum bandwidth (e.g., Category A). However, if no QoS queues meet this criteria, Category B is selected.

At step 430, if multiple QoS queues are members of the selected QoS category, processing continues with step 440; otherwise, processing branches to step 470.

At step 440, the relative priorities of the QoS queues are used to select among the QoS queues of the selected category that have pending data.

At step 450, if two or more QoS queues have the same priority, then processing continues with step 460. Otherwise, if a QoS queue is found to have the highest relative priority, then processing branches to step 470.

At step 460, the tie is resolved by performing round robin or LRU scheduling. That is, until the QoS categories are evaluated, the QoS queues having the same priority will be rotated through in a predetermined order or scheduled such that the QoS queue that has not provided a

packet for transmission recently will be given such an opportunity. After selecting a QoS queue in this manner, processing continues with step 470.

At step 470, a packet is dequeued from the selected QoS queue and the packet is transmitted by the port at step 480. This scheduling process may be repeated by looping back to step 410, as illustrated.

Queuing Schemes

A variety of different queuing mechanisms may be implemented using various combinations of the QoS profile attributes discussed above. Table 2 below illustrates how to achieve exemplary queuing mechanisms and corresponding configurations of the QoS profile attributes.

Queuing Mechanism Configurations

Queuing Mechanism	QoS Profile Attribute Value
Strict Priority Queuing	Minimum Bandwidth = 0 % Maximum Bandwidth = 100 % Peak Bandwidth = 100 % Maximum Delay = N/A Relative Priority = $PRIORITY_i$
Round Robin/ Least Recently Used Queuing	Minimum Bandwidth = 0 % Maximum Bandwidth = 100 % Peak Bandwidth = 100 % Maximum Delay = N/A Relative Priority = <same for all queues>
Weighted Fair Queuing	Minimum Bandwidth = >0 % Maximum Bandwidth = MAX_BW_i Peak Bandwidth = $PEAK\_BW_i$ Maximum Delay = N/A Relative Priority = <same for all queues>

Table 2

$PRIORITY_i$ represents a programmable priority value for a particular QoS queue, i. Similarly, MAX_BW_i and $PEAK\_BW_i$ represent programmable maximum bandwidths and peak bandwidths, respectively, for a particular QoS queue, i.

For a strict priority scheme, each QoS queue's minimum bandwidth is set to zero percent, each QoS queue's maximum bandwidth is set to one hundred percent, and each QoS queue's peak is set to one hundred percent. In this manner, the current bandwidth will never be less than the minimum bandwidth, and the current bandwidth will never exceed either the peak bandwidth or the maximum bandwidth. In this configuration, all QoS queues will be associated with Category B since no QoS queues will satisfy the criteria of either Category A or Category B. Ultimately, by configuring the QoS profile attributes in this manner, the second level of arbitration (e.g., the relative priority of the QoS queues) determines which QoS queue is to source the next packet.

For a pure round robin or least recently used (LRU) scheme, the QoS profile attributes are as above, but additionally all QoS queue priorities are set to the same value. In this manner, the third level of arbitration determines which QoS queue is to source the next packet.

Finally, weighted fair queuing can be achieved by assigning, at least, a value greater than zero percent to the desired minimum bandwidth. By assigning a value greater than zero to the minimum bandwidth parameter, the particular QoS queue is assured to get at least that amount of bandwidth on average because the QoS queue will be associated with Category A until at least its minimum bandwidth is satisfied. Additionally, different combinations of values may be assigned to the peak and maximum bandwidths to prevent a particular QoS queue from monopolizing the link.

Alternative Embodiments

While evaluation of QoS categories has been described above as occurring periodically, this evaluation may also be triggered by the occurrence of a predetermined event. Alternatively, evaluation of QoS categories may take place as part of the scheduling processing rather than as part of a separate periodic background process.

While a relationship between the number of priority levels and the number of QoS queues has been suggested above, it is appreciated that the number of QoS queues may be determined independently of the number of priority levels. Further, it is appreciated that the number of QoS queues provided at each port may be fixed for every port or alternatively a variable number of QoS queues may be provided for each port.

Finally, in alternative embodiments, weighting factors and ratios other than those suggested herein may be used to adjust the current bandwidth calculation for a particular implementation.

In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

CLAIMS

What is claimed is:

1. A method of bandwidth management for use in a packet forwarding device coupled to a network, the method comprising the steps of:

receiving at a packet forwarding device information indicative of one or more traffic groups;

receiving at the packet forwarding device one or more bandwidth parameters for at least one of the one or more traffic groups;

receiving at a first port of a plurality of ports a packet associated with the at least one traffic group; and

scheduling the packet for transmission from a second port of the plurality of ports based upon the one or more bandwidth parameters for the at least one traffic group with which the packet is associated.

2. The method of claim 1, wherein the network employs a non-deterministic access protocol.

3. The method of claim 2, wherein the non-deterministic access protocol is Carrier Sense Multiple Access with Collision Detection (CSMA/CD).

4. The method of claim 1, wherein the one or more bandwidth parameters include a minimum bandwidth.

5. The method of claim 1, wherein the one or more bandwidth parameters include a maximum bandwidth.

6. The method of claim 1, wherein the one or more bandwidth parameters include a peak bandwidth.

7. The method of claim 1, wherein the one or more bandwidth parameters include a maximum delay.

8. The method of claim 1, wherein the one or more bandwidth parameters include a relative priority.

9. The method of claim 1, further comprising the steps of:
classifying the packet as being associated with the at least one traffic group; and
determining a quality of service queue with which the at least one traffic group is associated.

10. The method of claim 1, further comprising the steps of:
enqueuing the packet onto a queue associated with the traffic group;
determining a current bandwidth metric for the queue; and
dequeuing the packet from the queue if the current bandwidth metric meets a predetermined relationship with the one or more bandwidth parameters.

11. The method of claim 10, wherein the current bandwidth metric is evaluated periodically at the expiration of a predetermined time period, and wherein the step of determining a current bandwidth metric for the queue further comprises the steps of:
determining an actual bandwidth for a prior time period;
determining a bandwidth metric for the prior time period; and

combining a portion of the actual bandwidth for the prior time period with a portion of the bandwidth metric for the prior time period to arrive at the current bandwidth metric.

12. A method of bandwidth management and traffic prioritization for use in a network of devices, the method comprising the steps of:

defining at a packet forwarding device information indicative of one or more traffic groups;

defining at the packet forwarding device information indicative of a quality of service (QoS) policy for at least one of the one or more traffic groups;

receiving a packet at a first port of a plurality of ports;

identifying a first traffic group of the one or more traffic groups with which the packet is associated; and

scheduling the packet for transmission from a second port of the plurality of ports based upon the QoS policy for the first traffic group, and wherein the scheduling is independent of end-to-end signaling.

13. The method of claim 12, wherein the network of devices employs a non-deterministic access protocol.

14. The method of claim 13, wherein the non-deterministic access protocol is Carrier Sense Multiple Access with Collision Detection (CSMA/CD).

15. The method of claim 12, further comprising the steps of:

providing a plurality of QoS queues; and

mapping the first traffic group to a first QoS queue of the plurality of QoS queues.

16. The method of claim 15, further comprising the steps of:

determining a current bandwidth metric for each of the plurality of QoS queues;

dividing the plurality of QoS queues into at least a first group and a second group based upon the current bandwidth metrics and a minimum bandwidth requirement associated with each of the plurality of QoS queues;

if the first group includes at least one QoS queue, then transmitting a packet from the at least one QoS queue; otherwise transmitting a packet from a QoS queue associated with the second group.

17. A method of bandwidth management and traffic prioritization for use in a network of devices, the method comprising the steps of:

receiving at a packet forwarding device information indicative of one or more traffic groups;

receiving at the packet forwarding device information defining a quality of service (QoS) policy for at least one of the one or more traffic groups, the QoS policy including at least a minimum bandwidth;

providing a plurality of queues at each of a plurality of output ports;

associating the one or more traffic groups with the plurality of queues based upon the minimum bandwidth; and

scheduling a packet for transmission from one of the plurality of queues onto the network.

18. The method of claim 17, wherein the information indicative of the one or more traffic groups includes Internet Protocol (IP) subnet membership.

19. The method of claim 18, wherein the information indicative of the one or more traffic groups includes a media access control (MAC) address.

20. The method of claim 17, wherein the information indicative of the one or more traffic groups includes a virtual local area network (VLAN) identifier.

21. A method of bandwidth management and traffic prioritization for use in a network of devices, the method comprising the steps of:

providing a plurality of quality of service (QoS) queues at each of a plurality of output ports, each of the plurality of QoS queues associated with a minimum queue bandwidth requirement;

adding a packet to one of the plurality of QoS queues based upon a traffic group with which the packet is associated; and

scheduling a next packet for transmission onto the network from one of the plurality of QoS queues at a particular output port of the plurality of output ports by:

determining a current bandwidth metric for each of the plurality of QoS queues,

dividing the plurality of QoS queues into at least a first group and a second group based upon the current bandwidth metrics and the minimum queue bandwidth requirements, and

if at least one QoS queue of the plurality of QoS queues, so divided, is associated with the first group, then transmitting a packet from the at least one QoS queue; otherwise transmitting a packet from a QoS queue of the plurality of QoS queues associated with the second group.

22. The method of claim 21, wherein the current bandwidth for a particular QoS queue is calculated as follows:

$CURR\_BW_i = W1 \times CURR\_BW_i + W2 \times ACT\_BW_i$;

where:

$CURR\_BW_i$ represents the current bandwidth for a particular QoS queue,

W1 represents a first weighting factor,

W2 represents a second weighting factor, and

ACT_BW_i represents the actual bandwidth received by the particular QoS queue in a previous time interval.

23. The method of claim 22, wherein $W1 = (W-1)/W$, $W2 = 1/W$, and the previous time interval is the most recent time interval.

24. The method of claim 21, further comprising the step of selecting among QoS queues in the same group based upon relative queue priorities associated with the QoS queues.

25. The method of claim 21, further comprising the step of selecting among QoS queues in the same group based upon a round robin selection scheme.

26. The method of claim 21, further comprising the step of selecting among QoS queues in the same group based upon a least recently used (LRU) selection scheme.

27. The method of claim 21, wherein the first group comprises QoS queues associated with a minimum queue bandwidth requirement that is less than the corresponding QoS queue's current bandwidth metric, and wherein the second group comprises QoS queues associated with a minimum queue bandwidth requirement that is greater than or equal to the corresponding QoS queue's current bandwidth metric.

28. A method of bandwidth management for use in a packet forwarding device participating in a connectionless network, the method comprising the steps of:

receiving at a packet forwarding device information indicative of one or more traffic groups;

receiving at the packet forwarding device one or more bandwidth parameters for at least one of the one or more traffic groups;

receiving at a first port of a plurality of ports a packet associated with the at least one traffic group; and

scheduling the packet for transmission from a second port of the plurality of ports based upon the one or more bandwidth parameters for the traffic group with which the packet is associated.

29. A packet forwarding device for use in a network employing a non-deterministic assess protocol, the packet forwarding device comprising:

an input unit configured to receive information defining one or more traffic groups and one or more associated bandwidth parameters;

a plurality of ports configured to transmit packets onto an attached network segment, each port having a plurality of queues and configured to select a queue of the plurality of queue from which to transmit a next packet based upon the one or more bandwidth parameters; and

a filtering and forwarding engine coupled to the plurality of ports and configured to process received packets, the filtering and forwarding engine identifying a traffic group of the one or more traffic groups with which a received packet is associated and queuing the received packet for transmission from one of the plurality of ports based upon the identified traffic group.

30. A packet forwarding device for use in a network employing a non-deterministic assess protocol, the packet forwarding device comprising:

a filtering and forwarding engine configured to forward received packets based upon a traffic group with which the packet is associated; and

a plurality of ports coupled to the filtering and forwarding engine, each port of the plurality of ports configured to receive packets from the filtering and forwarding engine, each port of the plurality of ports having a plurality of Quality of Service (QoS) queues associated with a minimum queue bandwidth requirement, each port of the plurality of ports further configured to schedule a packet for transmission onto the network by

determining a current bandwidth metric for each of the plurality of QoS queues,

dividing the plurality of QoS queues into at least a first group and a second group based upon the current bandwidth metrics and the minimum queue bandwidth requirements, and

if at least one QoS queue of the plurality of QoS queues, so divided, is associated with the first group, then transmitting a packet from the at least one QoS queue; otherwise transmitting a packet from a QoS queue of the plurality of QoS queues associated with the second group.

31. The packet forwarding device of claim 30, wherein the plurality of ports are further configured to select among QoS queues in the same group based upon relative queue priorities associated with the QoS queues.

32. The packet forwarding device of claim 30, wherein the plurality of ports are further configured to select among QoS queues in the same group based upon a round robin selection scheme.

33. The packet forwarding device of claim 30, wherein the plurality of ports are further configured to select among QoS queues in the same group based upon a least recently used (LRU) selection scheme.

34. The packet forwarding device of claim 30, wherein the first group comprises QoS queues associated with a minimum queue bandwidth requirement that is less than the corresponding QoS queue's current bandwidth metric, and wherein the second group comprises QoS queues associated with a minimum queue bandwidth requirement that is greater than or equal to the corresponding QoS queue's current bandwidth metric.

35. A machine-readable medium having stored thereon data representing sequences of instructions, said sequences of instructions which, when executed by a processor, cause said processor to perform the steps of:

receiving at a packet forwarding device information indicative of one or more traffic groups;

receiving at the packet forwarding device information defining a quality of service (QoS) policy for at least one of the one or more traffic groups;

receiving a packet at a first port of a plurality of ports;

identifying a first traffic group of the one or more traffic groups with which the packet is associated; and

scheduling the packet for transmission from a second port of the plurality of ports based upon the QoS policy for the first traffic group, and wherein the scheduling is independent of end-to-end signaling.

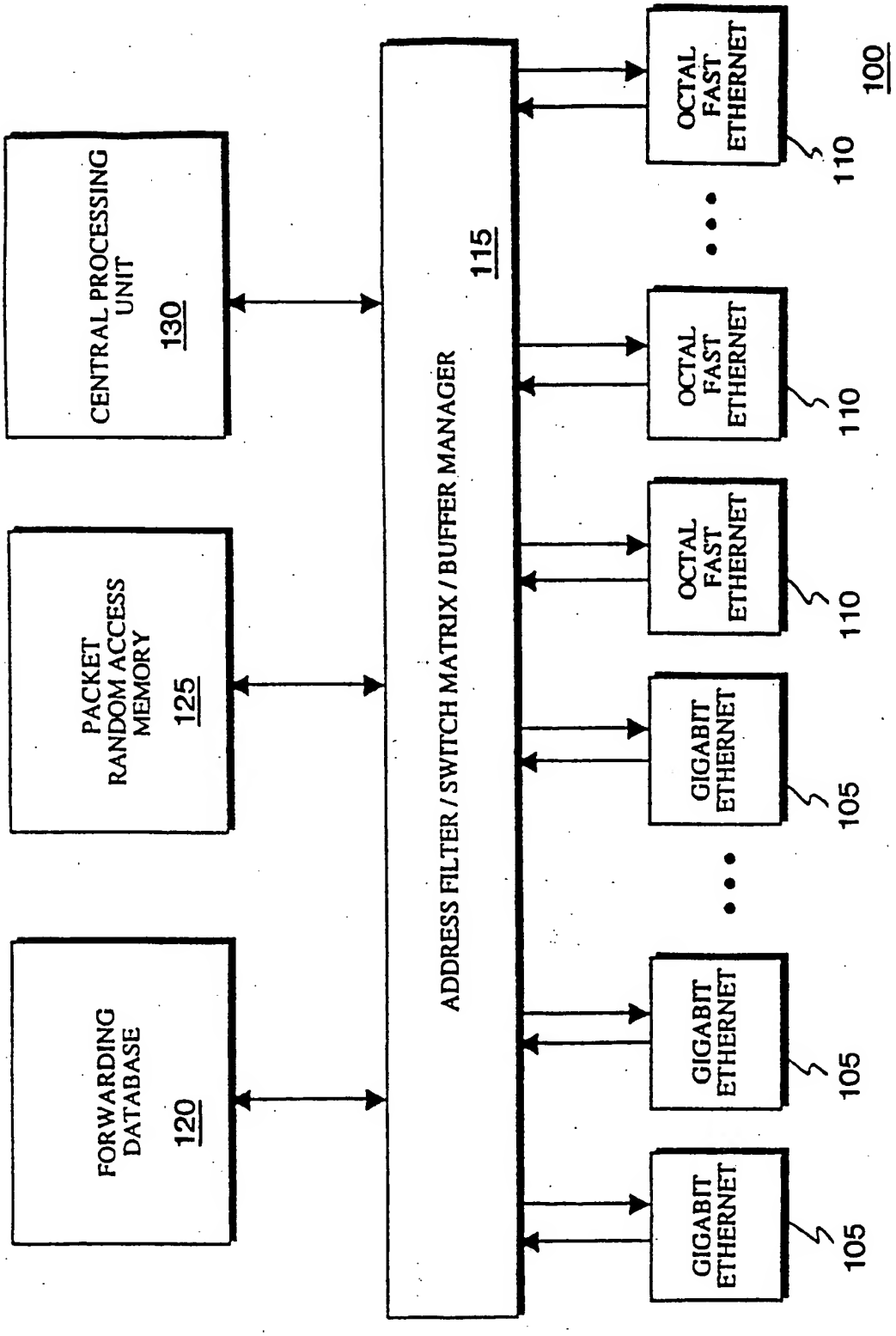

CENTRAL PROCESSING UNIT
130
PACKET RANDOM ACCESS MEMORY
125
FORWARDING DATABASE
120
ADDRESS FILTER / SWITCH MATRIX / BUFFER MANAGER
115
OCTAL FAST ETHERNET
110
GIGABIT ETHERNET
105
100

FIG. 1A

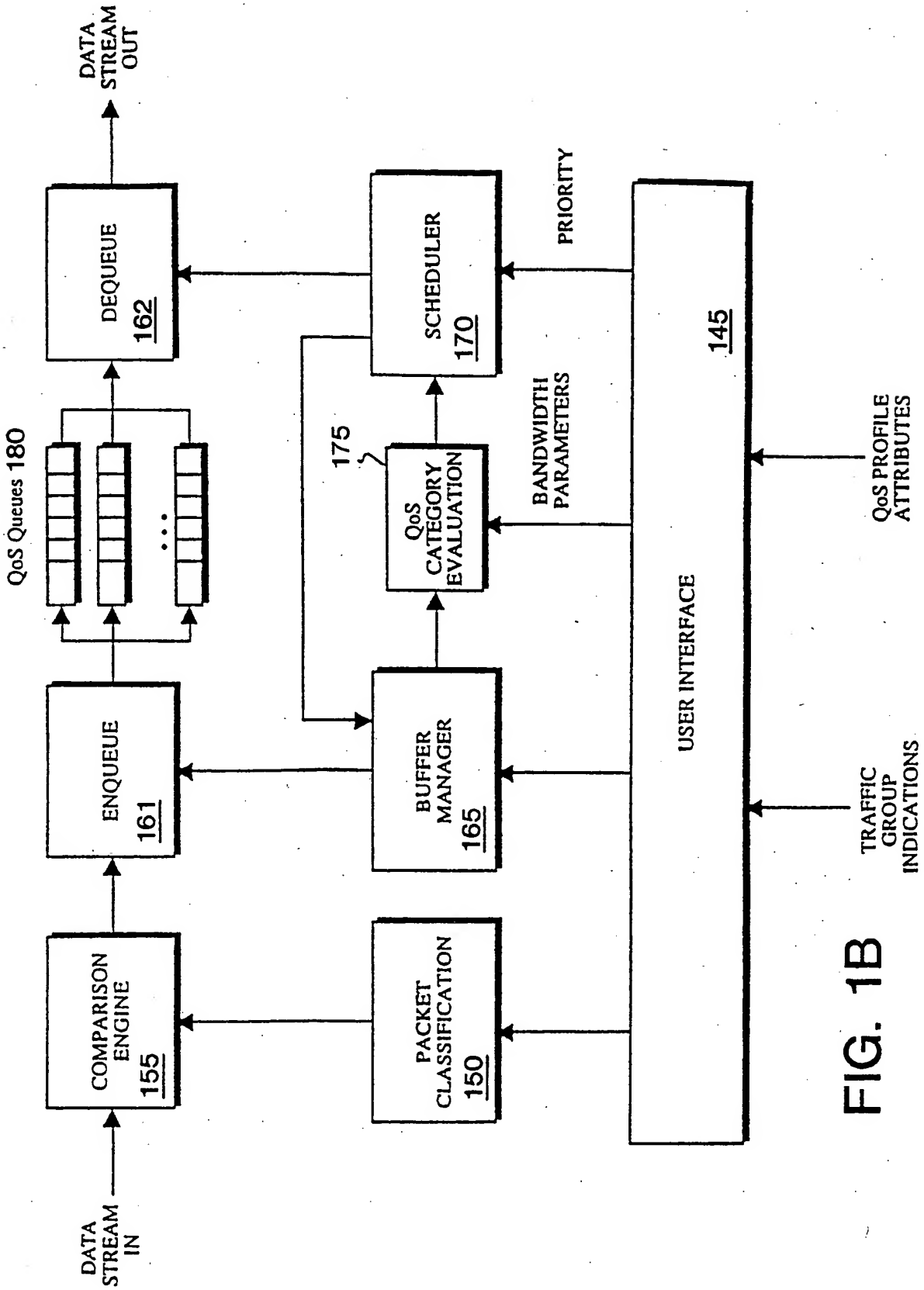

DATA STREAM OUT
DEQUEUE 162
SCHEDULER 170
PRIORITY
QoS Queues 180
175
QoS CATEGORY EVALUATION
BANDWIDTH PARAMETERS
145
QoS PROFILE ATTRIBUTES
ENQUEUE 161
BUFFER MANAGER 165
USER INTERFACE
TRAFFIC GROUP INDICATIONS
COMPARISON ENGINE 155
PACKET CLASSIFICATION 150
DATA STREAM IN

FIG. 1B

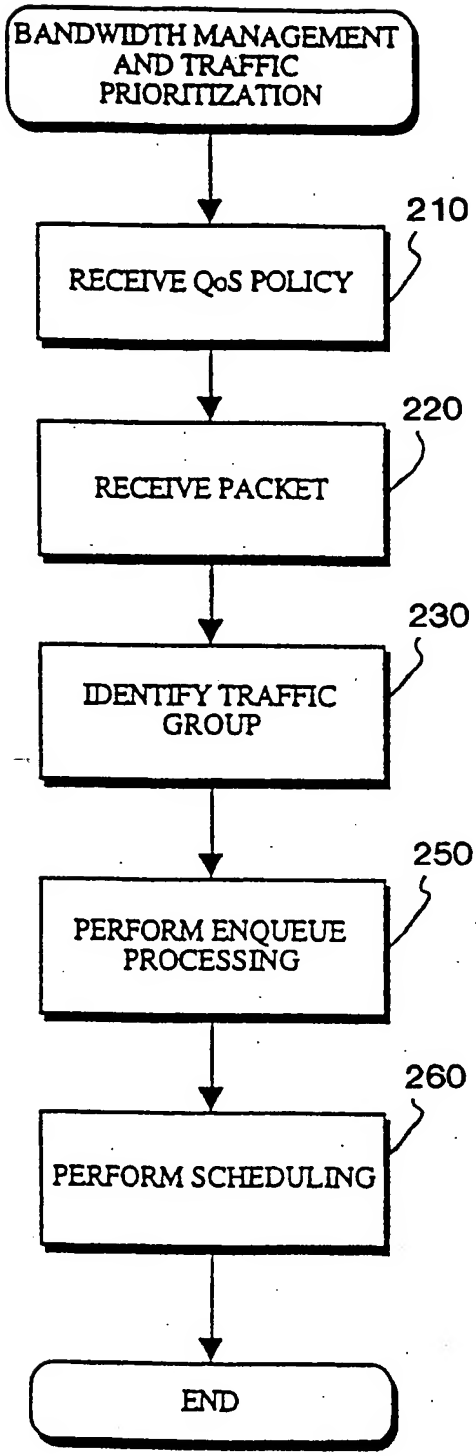

BANDWIDTH MANAGEMENT AND TRAFFIC PRIORITIZATION
210
RECEIVE QoS POLICY
220
RECEIVE PACKET
230
IDENTIFY TRAFFIC GROUP
250
PERFORM ENQUEUE PROCESSING
260
PERFORM SCHEDULING
END

FIG. 2

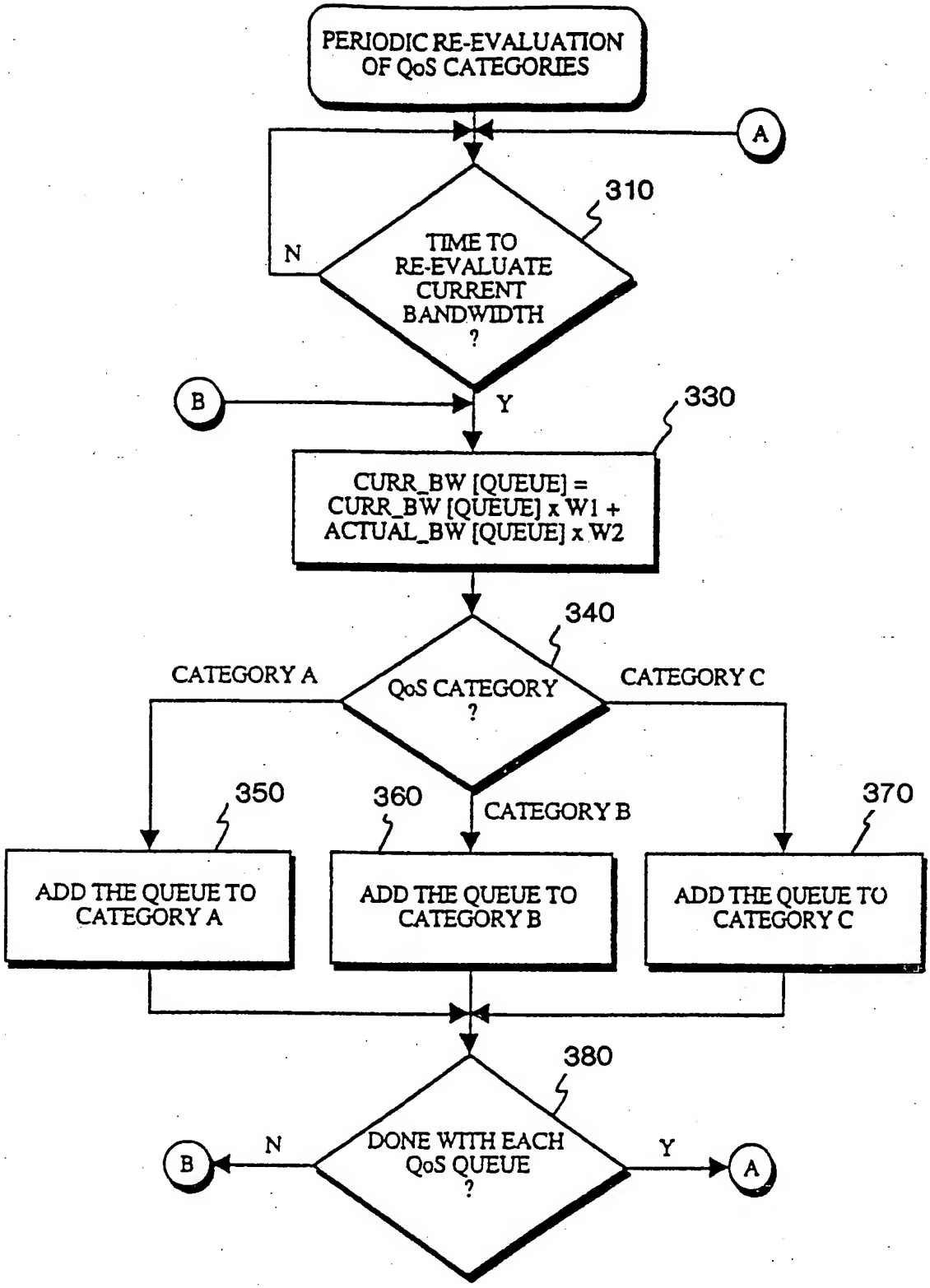

PERIODIC RE-EVALUATION OF QoS CATEGORIES
A
310
TIME TO RE-EVALUATE CURRENT BANDWIDTH ?
N
B
Y
330
CURR_BW [QUEUE] = CURR_BW [QUEUE] x W1 + ACTUAL_BW [QUEUE] x W2
340
QoS CATEGORY ?
CATEGORY A
CATEGORY B
CATEGORY C
350
360
370
ADD THE QUEUE TO CATEGORY A
ADD THE QUEUE TO CATEGORY B
ADD THE QUEUE TO CATEGORY C
380
DONE WITH EACH QoS QUEUE ?
N
Y
B
A

FIG. 3

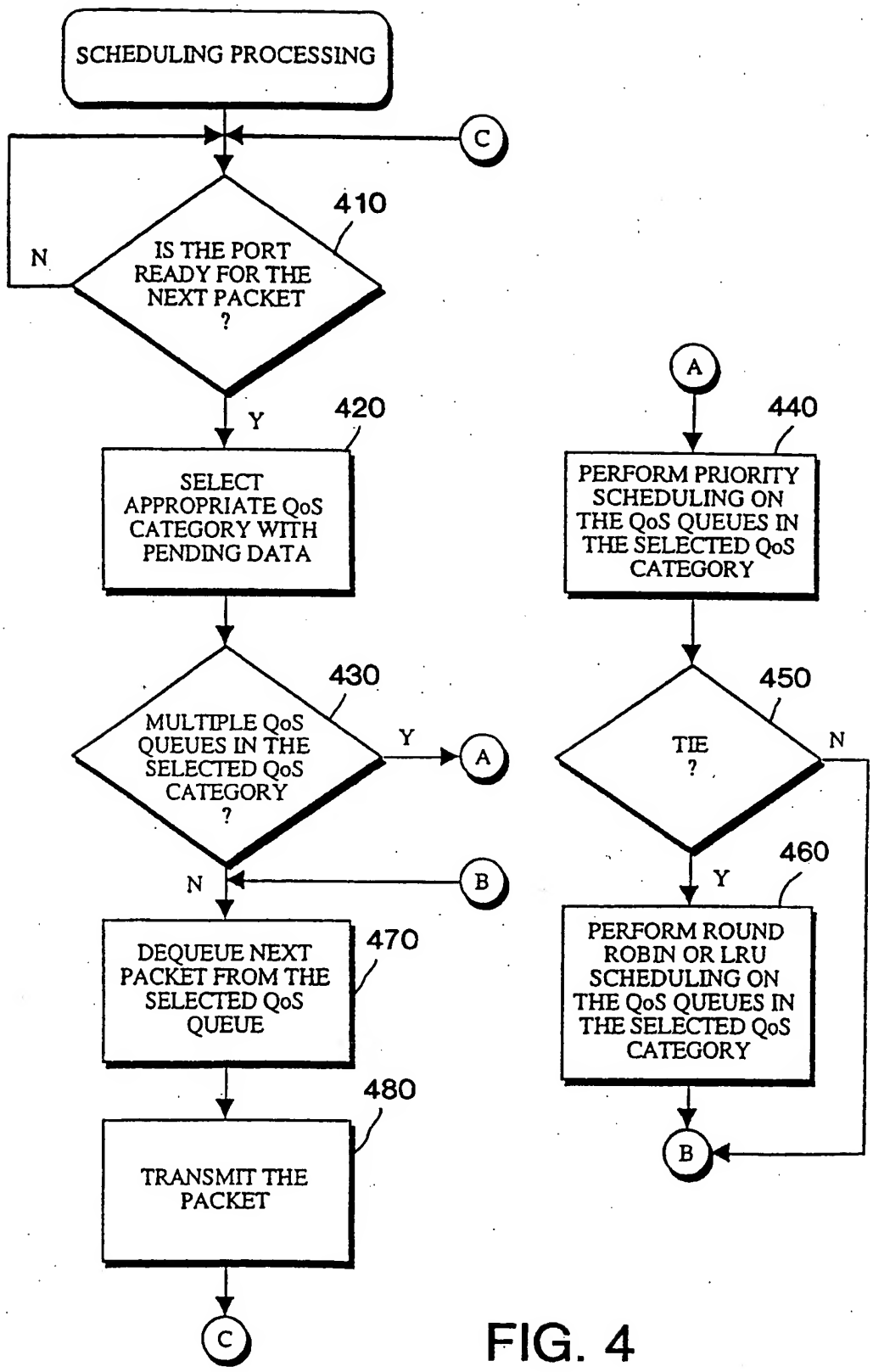

SCHEDULING PROCESSING
C
410
IS THE PORT READY FOR THE NEXT PACKET ?
N
Y
420
SELECT APPROPRIATE QoS CATEGORY WITH PENDING DATA
430
MULTIPLE QoS QUEUES IN THE SELECTED QoS CATEGORY ?
Y
A
N
B
470
DEQUEUE NEXT PACKET FROM THE SELECTED QoS QUEUE
480
TRANSMIT THE PACKET
C
A
440
PERFORM PRIORITY SCHEDULING ON THE QoS QUEUES IN THE SELECTED QoS CATEGORY
450
TIE ?
N
Y
460
PERFORM ROUND ROBIN OR LRU SCHEDULING ON THE QoS QUEUES IN THE SELECTED QoS CATEGORY
B

FIG. 4

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/17817

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :H04J 3/14
US CL :370/401

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 370/401, 229, 230, 231, 232, 233, 234, 235, 236, 252, 253, 400, 402, 410, 412, 413, 414, 415, 416, 417, 418, 445, 447

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,394,402 A (ROSS) 28 February 1995, figures 1, 5-7, col. 2, line 46 to col. 4, line 38.	1-15, 17-20 and 29
Y	US 5,499,238 A (SHON) 12 March 1996, figures 3a-5b, col. 2, line 35 to col. 3, line 49.	1-15, 17-20 and 29
Y, P	US 5,742,772 A (SREENAN) 21 April 1998, figures 6-7, abstract, col. 2, lines 2-35.	1-15, 17-20 and 29
Y,E	US 5,580,399 A (GANMUKHI et al) 15 December 1998, figures 1-2, col. 2, line 65 to col.5, line 30.	1-15, 17-20 and 29

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search
20 DECEMBER 1998

Date of mailing of the international search report
29 JAN 1999

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231
Facsimile No. (703) 305-3230

Authorized officer
HUY D. VU
Telephone No. (703) 308-6602

Form PCT/ISA/210 (second sheet)(July 1992)*

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04L 29/06 | A1

(11) International Publication Number: WO 99/01968

(43) International Publication Date: 14 January 1999 (14.01.99)

(21) International Application Number: PCT/US98/11928

(22) International Filing Date: 1 July 1998 (01.07.98)

(30) Priority Data:

08/886,869	2 July 1997 (02.07.97)	US
09/016,120	30 January 1998 (30.01.98)	US

(63) Related by Continuation (CON) or Continuation-in-Part (CIP) to Earlier Applications

US	08/886,869 (CIP)
Filed on	2 July 1997 (02.07.97)
US	09/016,120 (CIP)
Filed on	30 January 1998 (30.01.98)

(71) Applicant *(for all designated States except US)*: SITARA NETWORKS, INC. [US/US]; Suite 3, 60 Hickory Drive, Waltham, MA 02154 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: SRIDHAR, Manickam, R. [US/US]; 60 Juniper Road, Holliston, MA 01746 (US). BORUCHOVICH, Boris [US/US]; Apartment 13, 75 Page Road, Bedford, MA 01730 (US). SIGEL, Steven [US/US]; 352 Chestnut Street, North Andover, MA 01845 (US). LOUCHEZ, Sylvain [CA/CA]; 25 Regent Road, Wrentham, MA 02093 (US). KHAN, Malik, Z. [US/US]; 240 Western Avenue, Sherborn, MA 01770 (US).

(74) Agents: PRAHL, Eric, L. et al.; Fish & Richardson P.C., 225 Franklin Street, Boston, MA 02110-2804 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published

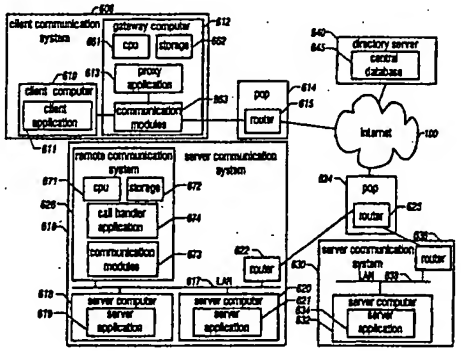
With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: ENHANCED NETWORK COMMUNICATION

(57) Abstract

A communication system in which client and server communication systems are coupled over a data network. In accordance with the invention, communication between the client and server communication systems has desirable characteristics, such as high throughput and low latency. Each of the server communication systems is configured to communicate with the client communication system using at least one of a set of transport layer protocols. The client communication system includes a transport layer module which implements the set of transport layer protocols. The client communication sytem also includes a layered communication module coupled to the transport layer module. The layered communication module includes a protocol selector for receiving a request to communicate with a requested one of the server communication systems and, using this request to communicate, choosing one the set of transport layer protocols for communication with the requested server system. The server communication system includes a communication application, coupled to a transport layer module, for maintaining a transport layer communication stream with each of a number of client communication systems, for accepting requests over the communication systems, for accepting requests over the communication streams from client communication systems to communicate with the one or more server applications, and for passing information between the cleint communication systems and the server applications over the communication streams.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

ENHANCED NETWORK COMMUNICATION

Background of the Invention

The invention relates to a distributed directory of information related to enhanced communication between computers coupled over a data network, such as enhanced communication between client and server computers coupled through the Internet.

The Internet has become a ubiquitous tool for accessing and retrieving information, and for conducting business in general. Accessing and displaying distributed linked multimedia documents on the Internet, known as browsing pages on the World Wide Web (the "Web"), has become an essential part of information retrieval both for business and pleasure. The Internet has brought previously hard to find information to everyone's fingertips. Devices such as commerce servers are now enabling business transactions to be conducted through the Internet. Due in part to the convenience of obtaining information and carrying out commercial transactions, people are joining the Internet community at a very rapid pace. This explosive growth of the number of users and the popularity of the available services has put a strain on the network which has become congested. This congestion has lead to users experiencing undue delays while trying to retrieve information and communicate through the network. The congestion also leads to the Internet behaving inconsistently. One can experience almost instantaneous response at certain times of the day, while it may appear to be impossible to reach the same server at other times of the day. Long delays and inconsistency diminish the user experience and may result in lost business opportunities.

Referring to Fig. 1, client and server computers C1-C9, S1-S4 (that is, computers executing the client and server applications) are coupled to the Internet 100. The Internet itself includes high speed ("backbone") data connections typically operating at data rates in the range of 45Mb/s (e.g., T3 capacity telephone trunks) or higher connected by switches or routers that forward packets towards their destinations. Computers C1-C9, S1-S4 are connected to the Internet through network Points of Presence (POPs) 110a-110d. A POP typically includes routers 112a-112d that are coupled to the Internet through data connections 114a-114d each having a capacity typically in the range of 1.5Mb/s (e.g., a T1 capacity telephone connection) to 45Mb/s (T3 capacity). Client computers C1-C3 can connect to a POP in a variety of ways, including those described below.

Client computers C1-C3 connect directly to a POP 110a over slow-speed, telephone modem connections 121-123 communicating a data rates in the range of 28kb/s to 56kb/s.

Clients computers C4-C6 are connected to each other within a single location using a local area network (LAN) 130 and a single computer or router serves as a gateway device 132. This gateway may serve a variety of functions, including packet routing, packet filtering (a security firewall), and various types of proxy service. The connection 124 between gateway device 132 and POP 110a is similar to that of the individual clients, although the data rate is typically higher, for example, in the range of 128kb/s (e.g., an ISDN telephone connection) to serve the requirements of the multiple clients.

Client computers C7-C9 connect directly to a POP 110b, but access a gateway device 140 at the POP that acts as a proxy server coupling the clients to a router

112b and then to the Internet. The connections 127-129 between the clients and the POP typically are slow-speed telephone modem connection. The connection between the client and the proxy server may use standard protocols or may use a proprietary protocol not generally used elsewhere in the Internet.

Servers S1-S4 are connected to POPs 110c-110d, although the communication capacity between a server site and a POP is typically 1.5Mb/s or higher. At the server sites, local area networks 150, 152 having a capacity of 10Mb/s or higher couple multiple servers and routers 154, 156 that are used to communicate with the POPs.

Internet communication is based on a layered model of communication protocols consistent with that published by the International Standards Organization (ISO) as shown in Fig. 2. The set of ISO protocol layers, or protocol stack, is numbered from one, at the lowest layer, to seven, at the application layer.

Communication over the Internet is based on packet-switching techniques. Addressing and transport of individual packets within the Internet is handled by the Internet Protocol (IP) corresponding to layer three, the network layer, of the ISO protocol stack. This layer provides a means for sending data packets from one host to another based on a uniform addressing plan where individual computers have unique host numbers and each computer has a logical set of numbered ports that can be addressed individually. By making use of the IP layer, a sending computer is relieved of the task of finding a route to the destination host. However, packets may be lost or damaged and are not guaranteed to be delivered in the order sent. Therefore, the sending host needs to make sure that the data sent is received successfully and that a series of individual packets is assembled appropriately.

A common denominator for the Internet is the "everything over IP" paradigm. There are protocol variations above layer three, for example, various application and transport protocols, and protocol variations below layer three, for example, various communication paths making up the network infrastructure, but layer three does not change. This allows IP to be the sole routing scheme in the Internet thereby enabling the worldwide connectivity which is a major ingredient of its success.

A transport layer protocol provides end-to-end communication between applications executing on different computers and regulates the flow of information between those applications. Rate and flow control are two examples of regulations of the flow of information. A transport layer protocol may also provide reliable transportation of information including, for example, in-sequence delivery of information and retransmission of lost or damaged information. Today, the Transmission Control Protocol (TCP) is used almost exclusively to provide end-to-end reliable (i.e., error free) data streams between computers over the Internet. TCP is layered on the IP protocol and corresponds to ISO layer four, the transport layer.

Software that supports the TCP protocol is provided on most popular operating systems, such as Microsoft Windows 95 and Windows NT, and most variants of Unix. An application using TCP is relieved of the details of creating or maintaining a reliable stream to a remote application by simply requesting that a TCP-based stream be established between itself and a specified remote system.

As a result of TCP being essentially universally accepted as the transport protocol, various client server applications have evolved which layer application-

specific protocols on top of end-to-end TCP communication channels, which are in turn layered on the IP network layer. Application layer protocols for file transfer, FTP (file transfer protocol), and for Web page access, HTTP (hyper-text transfer protocol), are two examples of popular application protocols layered on TCP.

The World Wide Web implements a system in which client applications, e.g., browsers such as Netscape Navigator or Microsoft Internet Explorer, can access and display linked documents, called Web pages, through server applications using the application layer hyper-text transfer protocol, HTTP. An address of a Web page or related data, referred to as a URL (uniform resource locator), typically includes a server host name and a symbolic reference to the data. The browser typically establishes a TCP-based connection to a predetermined port on the server host. That port is monitored by the server process. The client and the server communicate using the HTTP protocol over one or more TCP connections. Today, HTTP version 1.0 is commonly used.

A Web page typically includes references (URLs) to other files that also must be retrieved in order to complete the rendering of the originally requested page. A browser interprets incoming data from a server, determines the URL of other files that are needed, and establishes concurrent TCP connections to retrieve those subordinate files as well. The subordinate files do not necessarily come from the same server. For example, a scanned image included on a Web page, such as an advertising banner, will be included in that page as a reference to a separate file on a different server. Such a scanned image file is retrieved over its own TCP connection.

TCP based communication can use an end-to-end sliding window protocol where many packets of data can be

sent before requiring that data in the first packet is acknowledged by the receiver. If one packet is lost or damaged, the sender determines after a time-out period that the packet needs retransmission and the entire sequence must be restarted at the un-acknowledged packet in a "Go-Back-N" paradigm. The timeout period must be significantly greater than a typical round-trip time from one host to the other and back to avoid premature timeouts. All the packets sent after the lost or damaged packet are sent again. Since most of the packets sent after the lost or damaged packet have likely been received successfully, this error recovery procedure results in unnecessary use of communication capacity. There is no means for the receiver to simply request the missing packet using TCP. A very small window is generally used on channels with high rates of packet loss or error. A small window can result in low throughput.

Fig. 3 shows an exemplary sequence of data transfers between a representative client computer C1 and a representative server computer S1 using an end-to-end TCP channel over a communication path which is transported through POPs 110a and 110c and through the Internet 100, as shown in Fig. 1. Client computer C1 is represented in Fig. 3 by vertical line 302 and server computer S1 by vertical line 304. Time flows from top to bottom and each arrow represents a data packet traveling across the communication channel. For illustration, we assume that TCP is operating with a sliding window size of four packets. The client sends a request R1 to the server who sends back acknowledgment AR1. The server then sends a sequence of data packets D1-D4 and then must wait for an acknowledgment to D1 before proceeding. In this example, the server can start sending data as soon as it has receive the request. Acknowledgments AD1 and AD2 are received by the server who proceeds to send data

packets D5 and D6. For illustration, the sixth packet D6 is lost near the midpoint of the communication path. Data packets D7-D9 are transmitted after acknowledgments AD3-AD5 are received. The server now waits to receive acknowledgment for the lost sixth packet D6. After a time-out period 310, the server retransmits the sixth packet D6' and then continues in sequence with the retransmissions D7'-D9'.

Referring to Fig. 4, using HTTP to retrieve data for a Web page which includes embedded references to other data requires several TCP exchanges. Fig. 4 shows the sequence of data transfers (without showing the acknowledgments) in which client computer C1, represented by vertical line 402 requests and receives a Web page from server computer S1, represented by vertical line 404. No transmission errors are illustrated in this case. Acknowledgments are not shown. Client computer C1 sends a request G1 to server computer S1. Server computer S1 responds with Web page P1. The client computer parses page P1 and determines that it needs two additional documents and issues requests G2 and G3. Server computer S1 receives the requests and sends data P2 and P3 concurrently to the client computer.

Fig. 5 shows an exemplary sequence of data transfers between a representative client computer C4 that is serviced by a proxy application, hosted on a gateway computer 132, and a representative server computer S1 (Fig. 1). Client computer C4 is represented by vertical line 502, gateway computer 132 is represented by vertical line 504, and server computer S1 is represented by vertical line 506. Separate TCP channels are established between client computer C4 and gateway computer 132 and between the gateway computer and server computer S1. Communication between the client computer and the gateway computer uses TCP but encapsulates

application-specific requests and responses in a proxy protocol. The proxy application strips the proxy protocol from outbound packets and forwards them to the intended recipient. The proxy application therefore acts as a server from the point of view of the client application and acts as a client from the point of view of the server application. Inbound packets are received by the proxy application, wrapped with the proxy protocol and forwarded to client application. Client computer C4 sends a request G11 to gateway computer 132. Gateway computer 132 forwards the request as G12 to server computer S1. Server computer S1 responds with Web page P11 which is forwarded by gateway computer 132 to client computer C4 as P12. The client computer parses page P12 and determines that it needs two additional documents and issues requests G21 and G31 which are forwarded to server computer S1 as G22 and G32 by gateway computer 132. Server computer S1 receives the requests and sends the requested data concurrently to the gateway computer as P21 and P31. The gateway computer forward the data to the client computer as P22 and P32.

Referring to Fig. 1, a proxy application serving the same function as that hosted on gateway computer 132 described above can be hosted on proxy server 140. In this case, a sequence of data transfers between a representative client computer C7 that is serviced by a proxy server 140 at POP site 110b and a representative server S1 follows the same pattern as shown in Fig. 5. Although the sequence of transfers is the same, in the previous case the data rate between the client application and the proxy application is high and the connection between the proxy application and the Internet is slow, while in this case, the connection between the client application and the proxy application is slow and

the connection between the proxy application and the Internet is high.

Summary

In a general aspect, the invention provides desirable communication characteristics between applications executing on computers coupled over a data network. For instance, the invention can provide higher throughput and reduced latency between a client browser application and a Web server application coupled over the Internet.

In one aspect, in general, the invention is a communication system in which a client communication system is coupled to a data network for communicating with multiple server communication systems. Each of the server communication systems is configured to communicate with the client communication system using at least one of a set of transport layer protocols. The client communication system includes a transport layer module which implements the set of transport layer protocols. The client communication system also includes a layered communication module coupled to the transport layer module. The layered communication module includes a protocol selector for receiving a request to communicate with a requested one of the multiple server communication systems and, using this request to communicate, choosing one the set of transport layer protocols for communication with the requested server system. The client communication system can additionally include an application coupled to the layered communication module or the client system can include an application which includes the layered communication module. Also, the client communication system can include a client computer programmed to implement the transport layer module, the layered communication module, and the application.

The client communication system can include one or more of the following features.

The system can include a user interface module for accepting from a user requests to communicate with server communication systems and for presenting to the user information sent from the server communication system. For instance, the user interface module can be part of a Web browser application.

The application can include a server module coupled to multiple client applications over a local data network for accepting requests from the client applications to communicate with server communication systems and for passing information between the server communication systems and the client modules. For instance, this application can be a proxy server application.

The client communication system can include a client computer programmed to implement at least one of the client applications and include a proxy computer programmed to implement the transport layer module, the layered communication module, and the application. In this arrangement, the client computer and the proxy computer are coupled by the local data network. The client communication system can also include a second client computer coupled to the proxy computer over the local data network and programmed to implement another of the client applications.

The client communication system can include a server table coupled to the protocol selector of the layered communication module. The server table holds information related to the transport layer protocols with which server communication systems are configured to communicate. The server table can include a first table for holding information associating identifications of server communication systems, such as names or network

addresses, and information related to one or more of the transport layer protocols that the client communication system can use to communicate with each of the identified server communication systems. The information related to the transport layer protocols for a server communication system can include an address of a communication server computer in the server communication system coupled to the data network for communicating with the identified server communication system. The server table can also include a second table for holding information associating identifications of server communication systems and identifying one or more transport layer protocols with which the identified server communication systems are not configured to communicate. The client communication system can include a table interface module for providing a user interface for viewing and modifying information in the server table. The information in the server table related to the transport layer protocols can include information related to an expiration time after which some of the information related to the transport layer protocols is not longer valid.

The transport layer module in the client communication system can include a rate control module for negotiating a maximum rate of data transmission with server communication systems.

The layered communication module can include a data compression and decompression module for compressing and decompressing data communicated using at least one of the transport layer protocols. Also, the layered communication module can include a security module for encrypting and decrypting data communicated using at least one of the transport layer protocols.

At least one of the set of transport layer protocols supports selective retransmission and at least

one other of the transport layer protocols does not support selective retransmission.

The layered communication module can include a multiplexor and a demultiplexor for combining a number of data streams associated with separate requests to communicate with server communication systems into a smaller number of transport layer data streams for communicating with the server communication systems.

The client communication system can include a directory service module coupled to the protocol selector of the layered communication module for accessing, over the data network, information related to the transport layer protocols with which server communication systems are configured to communicate.

In another aspect, in general, the invention is a communication system which includes a server communication system which is coupled over a data communication network to multiple client communication systems. The server communication system includes a transport layer module for communicating with the client communication systems and one or more server applications. The server communication system also includes a communication application coupled to the transport layer module for maintaining a transport layer communication stream with each of a number of client communication systems, for accepting requests over the communication streams from client communication systems to communicate with the one or more server applications, and for passing information between the client communication systems and the server applications over the communication streams.

The server communication system can include one or more of the following features.

The server communication system can include a communication server computer coupled to the data

communication network and programmed to implement the transport layer module and the communication application. The communication server computer can be programmed to implement at least one of the server applications. Also, the server communication system can include an application server computer programmed to implement at least one of the server applications and a local data network coupling the communication server computer and the application server computer.

The transport layer module in the server communication system can implement a set of two or more transport layer protocols. The communication application is configured to communicate with the server applications using a first of the set of transport layer protocols and to communicate with client communication systems using a second of the set of transport layer protocols. For instance, the second transport layer protocol supports selective retransmission and the first transport layer protocol does not support selective retransmission.

The communication application can be configured to limit accepted requests from client communication systems to communicate with server applications based on a total allocated communication rate of accepted requests.

The server communication system can include a server table identifying server applications for which the communication application is configured to pass information to a client communication system. The server communication system can also include a module for accepting information to update the server table.

The communication application can be configured to pass information between more than one server application and a client communication system over a single transport layer communication stream.

The server communication system can include an address translation table for associating network

addresses provided by client communication systems as identifiers of server applications with local network addresses used for communicating between the communication application and the server applications.

The address translation table can be configured to associate more than one local network address for each network address provided by a client communication system, and the server communication system can include a server selection module for selecting one of the local addresses in response to a request to communicate from a client communication system. The server selection module can also include a table for storing an association between a client communication system and a selected local network address.

The transport layer module can include a rate control module configured to control a total rate of communication to each client communication system. Also, the communication application can include a compression module for compressing information passing from a server application to a client communication system. The communication application can include an encryption module for encrypting information passing from a server application to a client communication system.

In another aspect, in general, the invention is a communication system which includes a server communication system coupled over a data network to multiple client communication systems. The server communication system incudes a transport layer module for communicating with the client communication systems using multiple transport layer protocols, a server application, and a layered communication module coupled between the transport layer module and the server application. The layered communication module is for receiving requests from the server application to accept communication from the client communication systems, for maintaining a

transport layer communication stream with each of a number of the client communication systems, for accepting requests over the communication streams from client communication systems to communicate the server application, and for passing information between the client communication systems and the server application over the communication streams.

The server communication system can include a server computer coupled to the data network and programmed to implement the transport layer module, the server application, and the layered communication module.

In another aspect, in general, the invention is a method for communicating between a client communication system and multiple server communication systems over a data communication network. The method includes accepting a request to communicate including receiving an identification of one of the server communication systems. The method further includes using the identification of the server communication system to select one of multiple transport layer protocols for communicating with the server communication system, and then communicating with the server communication system over the data communication network using the selected transport layer protocol.

Selecting the transport layer protocol can include determining a set of one or more transport layer protocols for which the server communication system is configured to communicate. Determining the set of protocols can include accessing a server table and retrieving information related to the server communication system. Determining the set of protocols can also include updating the server table based on data received from the server communication system. Determining the set of protocols can include retrieving information related to the server communication system

from a directory computer over the data communication network. The address of the directory computer can be related to the identification of the server communication system. For instance, the address of the directory computer is the same as the address of a server computer identified in the request to communicate.

Accepting the request to communicate can include accepting a request to communicate with the server communication system using a requested transport layer protocol for which the server communication system is configured to communicate. The selected transport layer protocol can be different from the requested transport layer protocol.

Accepting the request to communicate with the server communication system can include accepting a request to communicate with a server computer at a first network address over the data communication network. The method can include selecting a second network address for communicating with the server communication system. Communicating with the server communication system can then include communicating with a computer at the second network address. The second network address can be different from the first network address.

The method can also include accepting a request for information through a user interface, providing a request to communicate with a server communication system based on the request for information, and providing the requested information through a user interface. Also, accepting the request to communicate can include accepting a request to communicate from a client application over a local data network.

In another aspect, in general, the invention is a method for communicating between a server communication system and multiple client communication systems over a data communication network. The method includes

establishing a first transport layer communication path between the server communication system and one of the client communication systems using a first transport layer protocol, receiving over the transport layer communication path a request from the client communication system to communicate with a first server application, establishing a second communication path to the first server application using a second transport layer protocol, and passing communication between the client communication system and the first server application over the first transport layer communication path and over the second transport layer communication path.

Prior to establishing the second communication path, the method can include determining whether to accept the request to communicate. The second communication path is then established only if the request is accepted.

Passing communication can include selectively retransmitting data not correctly received by the client communication system. Passing communication can also include limiting a total rate of communication with the client communication system.

The method can also include receiving over the first transport layer communication path a second request to communicate with a second server application, establishing a third transport layer communication path to the second server application using the second transport layer protocol, and passing communication between the client communication system and the second server application over the first transport layer communication path and over the third transport layer communication path. The first server application and the second server application can be the same but the second

transport layer communication path is separate from the third transport layer communication path.

Determining whether to accept the request to communicate can include accessing a server table identifying server applications for which communication should be passed. Determining whether to accept the request can also include determining whether accepting the request would exceed a total communication capacity.

The method can also include accessing an address translation table and determining a local network address of the first server application.

Aspects of the invention can include one or more of the following advantages. One advantage is improved throughput for communication between computers coupled over a data network. For instance, use of an alternative transport layer protocol on some or all of a communication path between two computers can improve throughput by using a selective retransmission strategy to correct errors.

Another advantage is that latency in fulfilling a request for data, such as a request for a Web page, can be reduced by reducing the number of exchanges needed to set up a connection to transfer the data.

The advantages of the invention do not necessarily require using modified client or server applications. For instance, the client and server applications can be configured to communicate with particular transport and application protocols, such as TCP and HTTP, while in fact, other transport and application layer protocols are used on some or all of the data path joining the applications.

Aspects of the invention can be applied to gateway and proxy computers, as well as to client and server computers which communicate directly with one another.

Other features and advantages of the invention will be apparent from the following description, and from the claims.

Description of the Drawings

Fig. 1 illustrates typical coupling of client and server computers to the Internet;

Fig. 2 shows the seven ISO communication protocol layers;

Fig. 3 shows an exemplary sequence of data transfers between a client computer and a server computer using TCP;

Fig. 4 shows an exemplary sequence of data transfers between a client computer and a server computer using HTTP;

Fig. 5 shows an exemplary sequence of data transfers between a client computer and a server computer communicating through a gateway computer using TCP;

Fig. 6 illustrates a client computer and server computers coupled to the Internet and shows a gateway computer and a remote communication server used for communication between the client computer and certain server computers;

Fig. 7 shows an exemplary sequence of data transfers between a client computer and a server computer through a gateway computer and a remote communication server;

Fig. 8 shows an exemplary sequence of data transfers between a client computer and a server computer using a modified HTTP protocol;

Fig. 9 shows an arrangement of software modules which execute on a gateway computer;

Fig. 9a shows an arrangement of software modules which execute on a client computer;

Fig. 10 is a flowchart of the operation of a redirector in response to requests from an application;

Fig. 11 is a flowchart of the operation of an HTTP Engine in response to requests from a redirector;

Fig. 12 shows an arrangement of software modules which execute on a remote communication server;

Fig. 13 shows an arrangement of software modules which execute on a server computer which supports communication using both TCP and XTP protocols;

Fig. 14 illustrates a client computer coupled over the Internet to a Domain Name Server, a list server, and a server LAN;

Fig. 15 shows an arrangement of software modules, including a directory module, on a client computer;

Fig. 16 shows a software modules on a client computer, including a detailed view of modules that are part of the directory module;

Fig. 17a shows the format of a database record for a server computer;

Fig. 17b shows the format of a record in a site file;

Fig. 18 is a flowchart of a host name resolution;

Fig. 19 is a flowchart of a layered service module handling a request to connect to a server computer;

Fig. 20 is a flowchart of a directory manager handling a request for a remote communication server address;

Fig. 21 is a flowchart of a directory manager retrieving remote communication server information and loadable modules;

Fig. 22 shows software modules, including loadable modules, in a layered service module; and

Fig. 23 shows a network address translator coupling a server computers and remote communication servers to the Internet.

Description

Embodiments of this invention involve communication between a client application and a server application over a data network, such as the Internet. An example of such communication is between a client application which is a Web browser and a server application which is a Web server, although other types of client and server applications can be involved as well. Furthermore, although one application is referred to as the "client" and one as the "server," embodiments of this invention are applicable to many situations when one application communicates with another over a data network and neither is exclusively a client or a server.

In the description that follows, a client application executes on a client computer which is coupled to the data network. A server application executes on a server computer also coupled to the data network. A server site includes one or more server computers on which server applications can execute, and in some embodiments of the invention, the server site also includes an additional computer used for communication between a client application and a server application executing at the server site. From the point of view of a client application, the combination of several computers at a server site can be viewed as a "server communication system" providing services to the client application. Similarly, from the point of view of a server application, a single client computer coupled to the Internet, or multiple client computers and a gateway or proxy computer can be viewed as a "client communication system" requesting services from the server application.

Several embodiments of the invention substantially share common functionality implemented in software modules executing on various computers, including client

and server computers as well as other computers, such as gateway computers, used for communicating between client and server computers. In a first embodiment of the invention, both the client communication system and the server communication system include multiple computers. Software modules which implement the common functionality are hosted on computers other than client or server computers which host the client and server applications. In a second embodiment software modules which implement the common functionality is hosted on the client and server computers themselves. Other embodiments use various combinations of computers to host software modules.

Referring to Fig. 6, a first illustrative embodiment of the invention supports communication between an exemplary client application 611 executing on a client computer 610 and exemplary server applications 619, 621, 634 executing on server computers 618, 620, 632 at server sites 616, 630. All the computers are coupled to Internet 100, which uses the Internet Protocol (IP) for network layer (ISO layer 3) communication. Client application 611 and server application 619,621,634 are configured to use TCP.

Client application 611 executing on client computer 610 communicates over the Internet with server computers 618, 620, 632 through a gateway computer 612 that in turn communicates with Internet 100 through POP 614. A proxy application 613 executes on gateway computer 612. Client application 611 is configured such that when it needs to establish a communication channel to a server computer, it contacts proxy application 613 with a request to establish that communication path. Once the communication channel is established further communication between client application 611 and the server computer passes through gateway computer 612 and

is handled by proxy application 613. From the point of view of a server computer, client computer 610 and gateway computer 612 function as a single client communication system 606. It appears to the server computer that an application on gateway computer 612, rather than an application on client computer 610, is requesting services. For example, the address of the client computer is generally not known by the server computer. Proxy application 613 can in general handle multiple communication channels between one or more client applications and one or more remote servers.

Two server communication systems 616, 630 include server computers 618, 620 at server communication system 616 and server 632 at server system 630 coupled to LANs 617 and 638 respectively. Routers 622 and 636 are coupled to LANs 617 and 638 respectively and provide access through POP 624 to Internet 100. Server applications 619, 621, 634 are hosted on server computers 618, 620, 632 respectively. Server communication system 616 is specially configured in that it also includes a remote communication server 626 (a computer) also coupled to LAN 617. Remote communication server 626 is used to pass certain communication between router 622 and server computers 618, 620. Server communication system 630 does not include a remote communication server computer.

In this first embodiment, communication between client application 611 hosted on client computer 610 and server application 619 hosted on server computer 618 at server communication system 616 can use two different types of transport layer communication paths. A first type of transport layer communication path is made up of two TCP-based segments in series, one between client computer 610 and gateway computer 612 executing a proxy application 613, and one between gateway computer 612 and server computer 618. The path followed by the second

segment passes from gateway computer 612 to router 615 at POP 614, through various communication links and routers in Internet 100, then to router 625 at POP 624, to router 622 on LAN 617 at server site 616, and finally to server computer 618.

Communication on the first segment between client computer 610 and gateway computer 612 uses TCP as the transport protocol. At the application layer, on the first segment, client application 611 communicates with proxy application 613 using a proxy protocol that incorporates application protocols used for the end-to-end application layer communication between the client application and the server application. On the second segment, proxy application 613 communicates with server application 619 using the appropriate application layer protocol for which the server application is configured. Two specific application protocols that are used to communicate between client application 611 and server applications are HTTP for accessing Web pages and data embedded in Web pages and FTP for accessing remotely stored files.

A second type of transport layer communication path between client application 611 on client computer 610 and server application 619 hosted on server computer 618 at server communication system 616 uses remote communication server 626 to forward communication between gateway computer 612 and server computer 618. Rather than communicating directly with server computer 618, gateway computer 612 communicates with remote communication server 626 which in turn communicates with server computer 618, thereby creating three separate segments on the path joining client computer 610 and server computer 618. The first segment is a direct path between client computer 610 and gateway computer 612. The second segment follows the path from gateway computer

612 to router 615 at POP 614, through various communication links and routers in Internet 100, then to router 625 at POP 624, to router 622 on LAN 617 at server site 616, and finally to remote communication server 626. The third segment is a direct path over LAN 617 between remote communication server 626 and server computer 618.

Communication on the first segment, from client computer 610 to gateway computer 612, uses the same protocols as on the same segment in the first transport layer communication path described above. Communication on the second segment joining gateway computer 612 and remote communication server 626, rather than using TCP, uses XTP, an alternate transport layer protocol on the second segment. Furthermore, when client application 611 and server application 619 are both using the HTTP application layer protocol, a data stream corresponding to that HTTP communication uses a modified HTTP protocol. Communication over the third segment from remote communication server 626 to server computer 618 uses TCP and standard application layer protocols including HTTP.

On this second type of transport layer communication path from client computer 610 to server computer 618 there are three segments at a transport layer (ISO layer 4). At the application layer (ISO layer 7) the communication path is made up of either one logical segment or three logical segments. When HTTP is not used, there is one logical segment joining the client and server application. That is, a sequence of data bytes sent by the client application are transported to the server application unmodified. When HTTP is used, there are three logical segments at the application layer. The first segment and the third segment use HTTP, while the second segment uses a modified HTTP protocol. Furthermore, a sequence of data bytes sent according the HTTP protocol from client application 611 is not

necessarily delivered to server application 619. HTTP data streams received at gateway computer 612 and at remote communication server 626 are interpreted and are not necessarily passed on without modification. Gateway computer 612 and remote communication server 626 cooperate to provide the needed translation into appropriate protocols for communicating with the client and server computers.

As there is no remote communication server at server communication system 630, communication between client application 611 at client computer 610 and server application 634 at server computer 632 uses a two-segment TCP-based communication path. The first segment is between client computer 610 and gateway computer 612 executing proxy application 613, and the second segment is between gateway computer 612 and server computer 632. The second segment passes from gateway computer 612 to router 615 at POP 614, through various communication links and routers in Internet 100, then to router 625 at POP 624, to router 636 on LAN 638 at server communication system 630, and finally to server computer 632. Gateway computer 612 can concurrently support communication directly with server computers as well as via remote communication servers.

When client application 611 initiates communication with a server application, such as server application 619 or server application 634, gateway computer 612 determines whether a data path through a remote communication server can be established, or whether a direct path to a server computer must be used. A path through a remote communication server is preferred since such a path can use the alternative transport and application layer protocols described above, which results in higher data throughput and lower latency than when using a direct path and standard transport and

application layer protocols to communicate between gateway computer 612 and a server computer.

Referring still to Fig. 6, gateway computer 612 includes CPU 661 and storage 662, such as a magnetic disk drive. Software stored in storage 662, when executed on CPU 661, includes proxy application 613 and communication modules 663. Communication modules 663 provide an interface for proxy application 613 to communicate with client application 611 and with server applications at the server sites coupled to Internet 100.

Remote communication server 626 includes CPU 671 and storage 672. Software stored in storage 672, when executed on CPU 671, includes call handler application 674 and communication modules 673. Communication modules 673 provide an interface for call handler application 674 to communicate with server applications 619 and 621 and proxy application 613.

Note that the term "module" generally is used to refer to a component of an operating system or an application, and "application" or "application program" is used to refer to a separate process managed by an operating system. As alternative embodiments can use different approaches to coordinate software components, the distinction between a component being a "module" or an "application" is not generally significant.

In this first embodiment illustrated in Fig. 6, a central database 645 is hosted on a directory server 640 also coupled to Internet 100. Database 645 includes information which can be used to identify a remote communication server which is be configured to communicate with a particular server computer. This database can be used by gateway computer 612 to determine whether a request to communicate with a server computer can be satisfied by establishing a communication path through a remote communication server. Each entry in the

database 645 associates a network address of a server application with certain information needed to set up an indirect path to that server application through a remote communication server. A network address of a server application includes a host address and port index of a port listened to by that server application. Information needed to set up an indirect path includes the network address used to connect to an appropriate remote communication server. In addition, the database can optionally be used to indicate that a particular application layer protocol is used by the server application at a particular server application address.

In this first embodiment, as introduced above, the transport (ISO layer 4) protocol used between gateway computer 612 and remote communication server 626 is based on the eXpress Transport Protocol (XTP). XTP is layered on the IP network protocol (ISO layer 3) which is used to route packets which make up the XTP communication between gateway computer 612 and remote communication server 626. XTP also supports bidirectional data communication over a single XTP connection.

XTP has several characteristics that differ from TCP and that give it advantages over TCP. One characteristic of XTP is that it supports use of a sliding window in combination with selective retransmission of lost or damaged packets. This combination allows efficient streaming of data over the XTP based segment joining gateway computer 612 and remote communication server 626.

Fig. 7 illustrates an exemplary sequence of data transfers involved in sending a request and receiving a multipacket reply along a communication path (Fig. 6) from client computer 610 through gateway computer 612 and remote communication server 626 and finally to server computer 618. For illustration, vertical lines 710, 712,

726, 718 in Fig. 7 represent client computer 610, gateway computer 612, remote communication server 626 and server computer 618, respectively, and diagonal lines illustrate data and acknowledgment packets that pass between the computers along the communication path. As described above, TCP is used on the first segment between client computer 610 and gateway computer 612 as well as on the third segment from remote communication server 626 and server computer 618. XTP is used on the second segment from gateway computer 612 to remote communication server 626. In this illustration, both the first and third, TCP, segments and the second, XTP, segment operate with a sliding window sizes of four packets and each packet is explicitly acknowledged.

A request R11 from client computer 610 is forwarded by gateway computer 612 as R12, and then forwarded by remote communication server 626 as R13. Acknowledgments AR11, AR12, AR13 are sent by gateway computer 612, remote communication server 626 and server compute 618, respectively, when the corresponding request packets are received. After acknowledging receipt of request R13, server computer 618 immediately begins sending data D11-D19. Remote communication server 626 has a large buffer for data packets and quickly accepts and acknowledges all the data packets from server 618. When remote communication server 626 receives the first data packet D11, it begins sending data D12 to gateway computer 612. This continues with data packets D22-D92. In this example, it is assumed that data packet D62 is lost at a point between the server computer and the client computer and is never acknowledged. Once the remote communication server determines that the packet is lost, either by a time-out or by an explicit negative acknowledgment (NACK), the remote communication server retransmits that packet as D62'. Note that since the

remote communication server has buffered the data and therefore does not have to request retransmission of the sixth packet from server 618. The gateway computer forwards data packets D12-D52 to client computer 610 as packets D13-D53 but waits for successful receipt of the sixth data packet D62' until it can deliver packets D63-D93 in the correct order to the client computer. Fig. 7 should be contrasted with Fig. 3 which illustrates a similar request, and reply on a single TCP connection. In Fig. 3, in addition to retransmitting the sixth packet, the seventh through ninth must be retransmitted as well. Also, since there is only one TCP segment, packet retransmissions must pass over the entire path from the server computer to the client computer and not simply over a portion of the path.

Other transport layer protocol characteristics in the XTP segment joining gateway computer 612 and remote communication server 626 include explicit rate control, which avoids congestion along a communication path, and multiplexing of multiple logical data streams between computers, which provides more efficient data transfer. Note that TCP does not have a similar explicit mechanism for rate control, and uses a separate instance of the TCP protocol for each logical data stream. As described more fully below, each of these characteristics yields performance advantages over using TCP.

With explicit rate control a sending computer can limit the rate at which data is sent along a communication path based on knowledge of the ability of the data path to transfer data. Referring to Fig. 6, consider the data path from remote communication server 626 and gateway computer 612. Along this path, data links of widely varying data rates are traversed. A 128kb/s link joins gateway computer 612 and POP 614 while a 10Mb/s link joins remote communication server 626 to

router 622. If remote communication server 626 sends data significantly faster than can be passed over the 128kb/s link from POP 614 to gateway computer 612, that data will have to be buffered somewhere along the path. This results in various inefficiencies including possible loss of a packet due to an overfull buffer, for example, a buffer at POP 614. Such a lost packet would only be discovered at the other end of the transport layer data stream, namely, at gateway computer 612 in this case. The lost data would then have to be retransmitted over the entire path. Rate control is used to limit the rate at which remote communication server 626 sends data to avoid this problem. In this case, the allowable rate of transmission from remote communication server 626 would not be significantly higher than the 128kb/s that can be sustained on the link from POP 614 to gateway computer 612.

Multiplexing enables a computer to use a single instance of the XTP protocol executing for a pair of computers communicating using XTP to handle multiple logical data streams between the two computers. This multiplexing capability is in contrast to TCP in which a separate instance of the TCP protocol executes independently for each logical data stream. An example of a situation in which multiple data streams are passing concurrently between two computers is when a Web browser requests data to render a particular Web page. If there are embedded references to other data in a Web page, separate TCP data streams, each with a separate instance of the TCP protocol, are used to retrieve the referenced data. Using XTP, if the data is retrieved from the same computer, the multiple data streams are multiplexed and use only a single instance of the protocol.

Bidirectional data communication using XTP enables one to implicitly open a reverse data channel when a

forward data channel is open. This is in contrast to TCP in which a reverse data channel must be set up using the same sequence of exchanges that are required to set up the forward data channel.

The previously mentioned modified HTTP protocol is used when a client application and a server application communicate using the HTTP protocol over an indirect communication path through a remote communication server. The modified HTTP protocol maintains the format of underlying data transported over HTTP (such as html formatted Web pages). In a first aspect of the modified HTTP protocol, multiple HTTP data streams between the client and the server are multiplexed on a single logical XTP data stream over the segment joining the remote communication server and the gateway computer. Note that multiplexing of multiple HTTP data streams onto one logical XTP data stream is different from and in addition to XTP itself multiplexing multiple logical data streams between a pair of computers for transmission using a single instance of the XTP protocol. Moreover, as is described further below, not all commands or data pass across the entire path from client application 611 to server application 619. For example, some client application commands send from the client application to the proxy application may be handled on the gateway computer and may not require services of the remote communication server or the server computer.

A second aspect of the modified HTTP protocol is that data is prefetched from server computers 618,621 by remote communication server 626 and is sent to the gateway computer 612 in anticipation of client application 611 making an explicit request for the data. The data is buffered at gateway computer 612 until it is requested by the client application. Remote communication server 626 determines what data to prefetch

based on references embedded in html format Web pages that are transferred from server computer 618 or 621 through the remote communication server to the client application.

Fig. 8 illustrates operation of the modified HTTP protocol which involves coordinated operation at the remote communication server and the gateway computer. This should be contrasted to a similar exchange using an end-to-end HTTP based connection shown in Fig. 4. Client computer 610, gateway computer 612, remote communication server 626, and server computer 618 (Fig. 6) are illustrated as vertical lines 810, 812, 826, 818, respectively, in Fig. 8. Transmission of Web page and other object requests and responses are shown as arrows with time increasing from top to bottom in the figure. Acknowledgments are not illustrated. Client computer 610 sends an initial "GET" request G11 for a Web page. Gateway computer 612 forwards the request from the client computer to remote communication server 626 as request G12. Remote communication server 626 receives G12 and requests the Web page from Web server 618 using a standard HTTP request G13. Web server 618 sends the requested page P11 to the remote communication server. Remote communication server 626 sends as page P12 to gateway computer 612, which in turn sends page P13 to client computer 610.

Remote communication server 626, in addition to forwarding page P11 received from server computer 618 to gateway computer 612 as page P12, interprets page P11 if it is in html format. Page P11 is parsed by an html parser and two embedded references to images or other objects found on that page are extracted. For illustration, two references in the received page P11 result in remote communication server 626 sending requests G23 and G33 to server computer 618. The server

computer responds with data P21 and P31 which are, in turn, forwarded by the remote communication server to gateway computer 612 as P22 and P32. When this data is received by the gateway computer, it is buffered since client computer has not yet requested the data. The data is effectively "prefetched" in anticipation of client application 611 on client computer 610 requesting that data. When gateway computer receives page P12 from remote communication server 626, it forwards that page as P13 to the client computer where it is interpreted by the client application that made the ordinal request G11. The client application makes requests G21 and G22 for the same data already requested by remote communication server 626 in requests G23 and G33. Gateway computer 612 does not forward the requests G21 and G31 since the data P22 and P32 which satisfies these requests has already been received and buffered by the gateway computer. The gateway computer passes the buffered data to the client computer. From the client computer's perspective, the fact that the data was prefetched is not evident other than in that the requests are satisfied with less delay than might be expected if requests G21 and G31 had been forwarded all the way to server computer 618 before being serviced.

There are at least two situations in which remote communication server 626 anticipates a request from client computer 610 and retrieves and sends the data to satisfy the request, but client computer 610 does not make the request as expected. The first situation is when the end user aborts retrieval of a Web page interactively with client application 611. In this case, client application 611 may never request the data referenced in the references embedded in received page. According to the HTTP protocol, an abort message is sent by client computer 610 to gateway computer 612 and this

abort message is forwarded to remote communication server 626. Once the remote communication server receives the abort message, further referenced data for that page is not sent. Data already sent to the gateway computer is buffered at the gateway computer but not forwarded to the client computer. In order that the buffer at gateway computer 612 does not grow too large, oldest unretrieved data is discarded by the gateway computer.

The second situation in which the requests are not made as expected for the embedded data is when the user "follows a link," that is, a user requests yet another page before the current page has been rendered and all embedded data has been received. In this case, no abort message is sent and all the data is prefetched. If the user returns to the original page, the embedded data will likely still be buffered on the gateway computer and the requests for that data can be satisfied without making another request of server computer 618.

Referring to Fig. 9, proxy application 613 and communication modules 663 executing on gateway computer 612 implement the functionality of the gateway computer as described above. This first embodiment uses the Microsoft Windows 95 or Windows NT operating system on the gateway computer. A description of the software modules that implement the functionality of remote communication server 626 follows the description of gateway computer 612.

Proxy application 613 interacts with several software modules in order to communicate with client computer 610, server computer 618, and other server computers and remote communication servers. Proxy application 613 can be implemented in a variety of ways, including those used in a number of commercially available proxy application programs. Typically, a proxy application has a server module 902 which accepts

requests from client applications executing on other computers, and a client module 904 coupled to the server module which communicates with the server systems.

In order to establish communication paths to client or server computers, proxy application 613 requests services from one or more communication software modules which implement various communication protocols. As normally configured in a typical installation of Windows 95 or Windows NT, TCP related requests from proxy application 613 would be passed directly to transport layer modules 940 which include TCP module 916. In this embodiment, a layered communication module 930 is coupled between proxy application 613 and transport layer modules 940. Layered communication module 930 includes a software interface module, a "hook," such that all TCP related requests from any application, and in particular from proxy application 613, are passed to redirector 914. The hook can be implemented as a layered service module within Winsock2. The redirector can pass these requests for TCP services to TCP module 916, to XTP module 956, or to HTTP Engine 920 which may request services from XTP module 956. TCP module 916 and XTP module 956 request services from Raw IP module 950 which in turn communicates with data and link layer module 952. Data and link layer module 952 is responsible for maintaining communication links with remote computers including client computer 610, server computer 618, and remote communication server 626.

Layered communication module 930 includes a compression module 918 for optionally compressing and decompressing data streams passing to and from transport layer modules 940, and a security module 917 for optionally encrypting and decrypting the data streams.

Not shown in Fig. 9 are additional software interface modules on the paths used to pass communication

requests from proxy application 613 to redirector 914, from redirector 914 to each of TCP module 916 and XTP module 956, and from HTTP Engine 920 to XTP module 956. These software interface modules accept requests according to the Windows Socket (Winsock) API as specified my Microsoft and pass the requests on to the respective modules. The software interface module on the path joining proxy application 613 and redirector 914 is implemented by the "hook" software interface module introduced above, and is configured to pass only TCP related requests from the proxy application to redirector 914. Requests by proxy application 613 for services involving other protocols than TCP are passed to other software modules which are not shown in the figure. The software interface module on the paths joining redirector 914 to TCP module 916 and to XTP module 956, as well as on the path joining HTTP Engine 920 to XTP module 956 use a Winsock2 module which is a dynamically linked library supplied by Microsoft. Winsock accepts requests according to the Winsock API and makes requests according to the Winsock Service Provider Interface (SPI).

Proxy application 613, as well as other modules using the Winsock API, request communication services in multistep sequences. These steps can include the following types of requests:

A. Request creation of a "socket" using a particular communication protocol, such as TCP or XTP. At any one time, this socket can be used for a single data stream. On successful completion of the request, a "handle" to the socket, an unsigned scalar index, is returned. Further requests related to this socket use the socket handle to identify the socket.

B. Request that an outbound communication channel be established to (connected to) a remote computer. For TCP/IP, the remote host address and port index are

specified as the terminating end of the communication channel.

C. Request that an inbound communication channel be established (listened for and accepted) from a remote computer on a particular port. The port may be the port already used for an outbound channel established in a step B above, or may be explicitly specified.

D. Send data on the outbound communication channel.

E. Receive data from the inbound communication channel.

Proxy application 613 makes a series of these communication requests specifying TCP as the communication protocol to be used. These requests are passed to redirector 914. In particular, in order to accept a connection from client computer 610 and then open a connection to server computer 618 on behalf of the client computer, client application 613 executes a series of communication requests including:

1. Create a socket (A) for communicating with client computer 610 using TCP.

2. Listen for and accept an inbound communication channel (C) on a particular port known to the client computer.

3. Request that an outbound communication channels be open (B) to the client computer. The port index at the client computer is the source port of the inbound communication channel.

4. Receive data (E) from the client computer. This data includes the address of server computer 618 with whom the client computer requests to communicate.

5. Create a second socket (A) for communicating using TCP.

6. Connect to server computer 618 (B) using the second socket.

7. Send data (D) (a request) received from client computer 610 to server computer 618.

8. Listen for an inbound channel (C) from server computer 618 on the port used for the outbound communication with the server computer.

9. Receive data (E) using the second socket from the server computer.

10. Send the received data (D) using the first socket to client computer 610.

Proxy application 613 makes the same of Winsock API requests regardless of whether a server computer is at a specially configured server site or a normally configured server site. The proxy application is not aware at the point of making the request whether a communication path through a remote communication server can be established, nor is it aware after communication has been established whether a direct TCP connection has been made to a server computer or whether an XTP connection has been made to a remote communication server.

At the application layer, when client application 611 communicates with server application 619 using HTTP, client application 611 creates an outbound data stream and receives an inbound data stream according to the HTTP protocol. When client application 611 sends HTTP data to proxy application 613, the proxy application requests that the data be written to an open socket but does not otherwise interpret it. Proxy application 613 makes the same request to write HTTP data regardless of whether it is communicating with a specially configured server site or a normally configured server site. The proxy application is not aware whether the HTTP data will be sent to server computer 618 over a TCP connection, sent first to remote communication server 626 using the modified HTTP protocol and XTP protocol, or handled on

the gateway computer without requiring communication with any other computer.

In the sequence of requests executed by proxy application 613 enumerated above, redirector 914 passes all the requests related to first socket, which is used to communicate with client computer 610, to TCP module 916. Communication between proxy application 613 and server application 619, hosted on server computer 618, over a direct path between gateway computer 612 and server computer 618 passes through redirector 914, TCP Module 916, and finally Raw IP module 950 and data and link layer module 952. Communication between proxy application 613 and a remote communication server passes through redirector 914, may pass through HTTP Engine 920, passes through XTP module 956, Raw IP module 950 and finally data and link layer module 952. XTP module 956 implements a similar level of functionality as TCP module 916 using XTP as the transport layer protocol rather than TCP. HTTP engine 920 interprets data streams passing through it and implements the client end of the modified HTTP protocol used on the communication segment between gateway computer 612 and remote communication server 626.

In order to determine whether an indirect communication path to a server computer can be established through a remote communication server, a protocol selector 923 in redirector 914 uses information obtained from central database 645 on directory server 640. This information is used to determine if a suitable remote communication server is available and if so, the address of that remote communication server. The host name or network address of directory server 640 is preconfigured in proxy application 613.

Rather than accessing central database 645 whenever it needs to establish a communication path to a server application, redirector 914 maintains server

tables 924 that reflect some of the information in central database 645. A first table, "in_table" 926, includes a subset of the entries in central database 645. If an entry is found in this table, the central database does not have to be queried since the information in central database 645 is available locally. A second table, "out_table" 928, includes addresses of server applications known to not have entries in central database 645. If a server application does not have an entry, that server application is accessed using a direct communication path between the gateway computer and a server computer. If a server address is found in out_table, there is no reason to query directory server 640 since it is known locally at gateway computer 612 that no entry will typically be found. These two tables are updated based on information in central database 645. A user interface application 912 is also coupled to server tables 924 to allow a user to view and modify information in the tables.

TCP module 916 receives calls from redirector 914 to open and communicate using the TCP protocols. TCP module 916 receives requests from redirector 914 using the Winsock SPI. When the TCP module 916 receives a request from redirector 914, the redirector is essentially transparent. A call to the TCP module is essentially identical to the call that would have occurred in a typical installation of Windows 95 or Windows NT in which all TCP requests are passed directly to the TCP module rather than to redirector 914.

TCP module 916 maintains socket data 931 which is used to store information about sockets it creates on behalf of applications such as proxy application 613. The socket data is used, for instance, to map a socket handle with an open data connection to a local port index and a remote host address and port index. In addition,

TCP module 916 includes data buffers 933 for connected inbound and outbound channels, and receiver and transmitter modules 936,948 used to implement the TCP protocol independently for each inbound or outbound connection. TCP module 916 communicates with Raw IP module 950, which implements the IP protocol layer, and which in turn communicates with a link and physical layer module 952. The link and physical layer modules is responsible for the communicating over the physical connections including those to client computer 610 and to router 615 at POP 614.

In addition to forwarding requests to TCP module 916, redirector 914 can also forward requests received from proxy application 613 to XTP module 956 and to HTTP Engine 920. Redirector 914 passes to XTP module 956 requests to open communication channels to and communicate with specially-configured server sites in the case that the data stream on that channel does not necessarily use the HTTP application layer protocol. Redirector 914 uses HTTP Engine 920 for HTTP based connections to specially configured server sites. Along with a TCP request, redirector 914 provides HTTP Engine 920 the TCP socket handle used by the client application and the address of a remote communication server that will receive the XTP communication.

XTP module 956 implements the XTP protocol. Logical data streams associated with XTP sockets are associated with XTP contexts. The logical structure of the XTP module is very similar to that of TCP module 916 except that all logical data streams to or from a particular host are multiplexed into a single data stream communicated using the XTP protocol whereas in the TCP module, each logical stream uses a separate instance of the TCP protocol. XTP module 956 includes data buffers 965 for connected inbound and outbound channels, and

receiver and transmitter modules 966, 976 used to implement the XTP protocol for each multiplexed data stream to a remote computer. XTP module 956 communicates with Raw IP module 950, which implements the IP network protocol layer, and which in turn communicates with a link and physical layer module 952. The link and physical layer module is responsible for the communicating over the physical connections including those to client computer 610 and to router 615 at POP 614. XTP module 956 maintains socket data 957 which is used to store information related to sockets created by the XTP module. Communication for multiple sockets between gateway computer 612 and a remote computer is multiplexed and demultiplexed by receiver and transmitter modules 966, 976 in XTP module 956 into a single inbound and a single inbound data stream and uses a single instance of the XTP protocol for each such stream. Socket data 957 is used to associate a socket handle with the local and remote port indices, as well as a key associated with the data stream associated with the socket. XTP module 956 includes a rate control module 977 for negotiating the data rate and then limiting the data rate to server systems.

An XTP based communication path between gateway computer 612 and a remote communication server is maintained for a period of time after all contexts are closed. If the client application tries to open a new connection to the remote communication server during this period, the connection is open with very little overhead. The period of time the connection persists, the "keep-alive time," can be a fixed interval or can be determined adaptively based on past communication characteristics.

Redirector 914 can also send a request to HTTP Engine 920 if it determines that a TCP request received from proxy application 613 corresponds to HTTP-based

communication to a specially-configured server site. HTTP Engine 920 interprets the application layer HTTP protocol used on a data connection. The HTTP Engine performs two functions in addition direct translation of TCP requests into XTP requests. First, the information in multiple HTTP data streams passing between the gateway computer and a particular remote communication server are multiplexed in HTTP multiplexor 982 for communicating using a single XTP context. Second, the HTTP Engine maintains prefetch buffers 984 which are used to service some HTTP requests for data. The HTTP multiplexor fills the prefetch buffers with data that has not yet been requested and provides the buffered data when a request can be satisfied with that data.

Redirector 914 maintains two additional data structures used in redirecting requests from proxy application 613 to the appropriate communication modules. When proxy application 613 requests creation of a TCP based socket, a TCP socket is indeed created for the proxy application and its handle is returned to the proxy application. At later point when proxy application 613 requests connection to a particular server computer, a second XTP based socket may be created at the request of redirector 914 if indirect communication with the server computer through a remote communication server is to be established. Redirector 914 maintains socket association table 915 which associates the TCP socket handle known to the proxy application and the XTP socket handle used for communicating with a remote communication server. Socket association table 915 also includes information needed to determine which communication module should handle requests for that socket.

Redirector 914 also includes a tracing buffer 927 used to record (trace) certain requests from proxy application 613 that are passed on to TCP module 916. In

particular, after the proxy application requests creation of a TCP socket, other requests related to that socket may be made by the proxy application prior to receiving a request to listen for an inbound connection or to connect to a particular remote computer. It is not until a request to establish a connection is received by redirector 914 that a determination can be made that the communication should use XTP rather than TCP. Therefore, these initial TCP requests are recorded in tracing buffer 927. If redirector 914 determines that an XTP socket should be created and associated with a previously created TCP socket, the recorded requests related to the TCP socket are "replayed" to the XTP socket. In this way, the XTP socket will be initialized such that proxy application cannot recognize that further requests directed to the TCP socket are now redirected to the new XTP socket. Alternatively, prior to determining whether XTP or TCP will be used, the requests can be processed in parallel using both protocols until a determination is made. Redirector 914 opens both a TCP and an XTP socket and sends communication requests to both sockets. Once the protocol is selected, the socket that will not be used further is destroyed.

Figs. 10 and 11 illustrate the detailed operation of redirector 914 and HTTP Engine 920. A detailed description of remote communication server 626 of this first embodiment follows the description of module operation in gateway computer 612.

Referring to Fig. 10, redirector 914 responds to a variety of requests proxy application 613. A request to create a TCP socket (1002) is passed to TCP module 916. A socket data handle is created by the TCP module and passed to the proxy application (step 1004).

If the request is to connect (that is, to open for writing) a TCP socket to a remote computer (1010), the

redirector first looks up the TCP socket handle in socket association table 915 (step 1011). If an XTP socket handle is associated with the TCP socket handle, the request is passed to the software module handling communication for that TCP socket (step 1013). If the TCP socket handle is not listed in the socket association table, the redirector looks up the host address in in_table 926 (step 1012). The in_table contains the addresses of servers that are known to be served by remote communication servers. If the address is not found (step 1014), then the request is forwarded to the TCP module (step 1016). Separately, either during or some time after the call to the TCP module, the redirector looks up the address in out_table 928 (step 1018). If the address is found (step 1020), then the addressed host is known to not be served by a remote communication server and no more processing is performed. If the address is not found in either the in_table or the out_table, the redirector accesses directory server 640 to update in_table and out_table (step 1022). If the address was found in the in_table (step 1014), then a remote communication server is servicing requests for the addressed host. The next step is to determine whether the addressed port on the addressed host is associated with an HTTP server (step 1024). This information is also stored in in_table along with the remote communication server address. If the connect request is to an HTTP server, a request to create a socket is passed to HTTP Engine 920. The HTTP Engine obtains an XTP socket handle from XTP module 956 and returns the socket handle to redirector 914. The redirector records the socket handle in socket association table 915 with the TCP socket handle used for the request by the proxy application, along with an indication that HTTP Engine 920 is now handling requests for that TCP socket. If the

addressed host is not an HTTP server, redirector 914 requests XTP module 956 to create a XTP socket (step 1027) and the TCP socket handle and the new XTP socket handle are recorded in socket association table 915. In the cases that a XTP socket is created by XTP module 956 or indirectly by HTTP Engine 920, requests recorded in tracing buffer 927 are replayed (step 1029) to the software module (XTP module 956 or HTTP Engine 920) now handling communication for the socket.

If the redirector receives a request other than one to create or connect a socket (1030) the redirector first looks up the TCP socket handle in socket association table 915 (step 1031). If an XTP socket handle is associated with the TCP socket handle, the request is passed to the software module handling communication for that TCP socket (step 1032) otherwise the request is sent to the TCP module (step 1034). Note that communication between proxy application 613 and client application 611 is established by the proxy application issuing a listen request using a TCP socket that the proxy application has associated with (bound to) a predefined port. This listen is passed to TCP module 916 according to step 1036.

Referring to Fig. 11, when HTTP Engine 920 receives a request to create an XTP socket to a remote communication server in order to service requests for a TCP socket to communicate with a server computer (step 1210), the HTTP Engine may use an already open XTP connection and multiplex communication for the TCP socket on the open connection. If there is no active XTP connection to the remote communication server (step 1212), HTTP Engine 920 requests creation of an XTP socket and connects to the remote communication server (step 1214). When the HTTP Engine receives a write request (step 1240), it parses the HTTP content of the request

(step 1242). If the request is to retrieve a remote object from the server (step 1244), the HTTP Engine first checks to see if the object is already in prefetch buffers 984 (step 1246). If it is, the HTTP Engine records the association of the TCP socket handle and the object requested (step 1248) so that subsequent listen and read requests can retrieve the appropriate buffered data. If the object is not buffered, the request is forwarded over the multiplexed data stream to the remote communication server (step 1250). If the write was not a request for an object (step 1244) the data is sent to the remote communication server (step 1252). When the HTTP Engine receives a request to listen on a connection that previously was used to send or record a request (step 1220), no further processing is necessary. When the HTTP Engine receives a request to read (step 1230), the object previously requested is determined from HTTP context 986. If all or some of the object is in prefetch buffer 984, that data is provided in response to the read request (step 1234). If there is no more data in the prefetch buffer (step 1232), for example if the transfer of the object was initiated before the request from the client, but is still in progress, the HTTP engine requests data from the XTP module using the appropriate XTP socket handle (step 1236). If the received data is for another object (step 1238), that data is stored in prefetch buffer 984 (step 1240), and another XTP read is requested (step 1236). If the received data is for the requested object, the data provided in response to the read request (step 1242).

In related embodiments, redirector 914 can also select a transport layer protocol based on other criteria. For example, a request to open a TCP connection to a port that is associated with a streaming audio server can be handled using a protocol different

from TCP or XTP that is well suited to the data being transferred. For example, a protocol with additional forward error correction can be used for streaming audio while a protocol with error control which relies on retransmission can be used for a non-streaming data source. The selection criterion can also be based on knowledge of the content type of the data. The content type can be determined in some cases by monitoring the initial portion of the data transmission.

The description above has concentrated on the functionality at gateway computer 612 which is part of the client communication system. At the server site, remote communication server 626 forms the endpoint of XTP-based communication with the gateway computer. Referring again to Fig. 6, remote communication server 626 acts as a gateway between gateway computer 612 and servers 618, 620. Together, remote communication server 626 and server computers 618, 620 form a single server communication system.

Referring to Fig. 12, call handler application 674 and communication modules 673 execute on a representative remote communication server 626. In the first embodiment, remote communication server 626 is a computer running a Windows NT or Unix based operating system. A standard TCP protocol stack including a TCP module 1314, an IP module 1316, and a link and data layer module 1318 are used to communicate to server computers 618, 620. An XTP module 1320 communicates directly with the IP module. Call handler application 674 communicates with communication modules 673 to handle communication between a gateway computer and server computers. In addition, the HTTP object prefetching function is implemented in call handler application 674.

Call handler application 674 includes a context handler module 1328 which directs communication between

gateway computers and server computers, html parser 1326 used to interpret html format data passing from a server computer to a gateway computer, local table 1322 which includes information about server computers served by the remote communication server, and TCP buffers 1324 used to hold data passing between server computers and gateway computers.

Context handler 1328 initially creates an XTP context and makes a listen request of the XTP module to accept a connection from a gateway computer. When a gateway computer connects an XTP context and requests to establish TCP communication with a destination network address, context handler 1328 looks up the destination network address for a server computer in a local table 1322 and, if it finds the destination network address, initiates an execution thread to handle communication with that gateway computer and the server computer. There is typically one execution thread per XTP context. The execution thread opens a TCP channel to the server application. A context thread may open multiple concurrent TCP channels to one or more server computers to handle multiplexed requests from its corresponding gateway computer. When the context handler is notified that a listen on a port has been requested by the client, a TCP listen is requested through TCP module 1314 and begins to read data that it buffers in TCP buffer 1324.

Context handler 1328 can optimally restrict connections between client and server computers. In one instance, context handler 1328 determines whether accepting a connection request from a client system would exceed a total communication capacity for the communication server computer. The communication capacity can be based on a variety of factors including the number of client systems, the number of TCP streams to server computers, or the maximum total data

communication rate as calculated by summing the negotiated maximum data rates of all the XTP connections or as calculated by averaging past actual data rates. If the communication capacity would be exceeded by accepting a connection, the connection is refused and the client system must connect to the server computer directly using TCP.

Alternatively, context handler 1328 can restrict connections between client and server computers based on the server computer identified in the communication request from the client computer. For instance, only server computers listed in local table 1322 can be accessed by a client computer. In this way, the remote communication functions as a firewall.

Context handler 1328 is also responsible for the server end of the modified HTTP protocol. As a Web (html format) page is retrieved from a Web server by a context handler through the TCP/IP stack, the page is parsed by html parser 1326. References to objects are extracted. The context handler then makes requests for the objects on servers also served by the remote communication server and forwards the results to the gateway computer which stores them in its prefetch buffer anticipating a request for them from the client application. Context handler 1328 also includes compression module 1329 and security module 1327 for providing compression and encryption services, respectively, for data streams passing between the remote communication server and the gateway computer.

Local table 1322 containing served hosts addresses is periodically communicated to a directory server 640 so that a gateway computer can locate an appropriate remote communication server for a TCP address. A table update module 1325 receives information from server computers to update local table 1322. For instance, the server computers can periodically send broadcast messages that

are received by table update module 1325, or the server computers can send a message to the table update module when they are initially configured. Alternatively, the addresses of the server computers can be entered manually through a user interface. Local table 1322 can also be updated manually through a graphical user interface (GUI) 1323 coupled to an input/output device 1324.

XTP module 1320 includes a rate control module 1321. Rate control module 1321 limits the total data rate to each client system rather than limiting the data rate of individual XTP connections. The rate limit for a particular client system is maximum of the negotiated maximum data rates of individual XTP connections to that client system. In this way, even if a client system opens multiple XTP streams, rate control module 1321 avoids exceeding the capacity of data links that may be shared by the multiple streams.

In a variant of the first embodiment, software modules that are hosted on a communication server computer and those hosted on a server computer are hosted together on a single computer. In this variant, communication passing between a call handler application and a server application does not flow across a local network. Instead, data is sent to communication services hosted on the single computer, and passed directly to the destination application by those communication servers.

A second embodiment of the invention implements the same functionality at the server system from the point of view of a client application and the same functionality at the client system from the point of view of a server application as in the first embodiment. However, in the second embodiment, software modules are hosted directly on a client computer or a server computer. In addition, since a gateway computer is not used, no proxy application is needed. The arrangement of

software modules on a client computer is shown in Fig. 9a. This arrangement is substantially the same as that shown for a proxy computer as shown in Fig. 9 with the exception that proxy application 613 is replaced with a client application 613a. Client application 613a can be a browser application which includes a user interface module 905 coupled to an input/output device 906 for accepting requests for information from a user and displaying information sent from server computers in response to the requests.

At the server system, in the second embodiment, the functionality of the server computer and the remote communication server are combined on a single computer which uses the Microsoft Windows 95, Windows NT, or Unix operating system. A server application 1412 configured to use TCP and HTTP executes on the server computer.

Referring to Fig. 13, the arrangement of software modules is similar to that shown in Fig. 9. Server application 1412 requests TCP services from layered communication services 1402. Redirector 1414 handles the request and communicates with transport services 1404, including TCP module 1416 and XTP module 1456, or HTTP Engine 1460 to handle services requested by server application 1412. Redirector maintains a socket association table 1415 that associates TCP socket handles created for the server application and XTP sockets created by the redirector.

Redirector 1414 initially receives a request from server application 1412 to listen on a predefined port for a TCP connection from a client system. Redirector 1414 determines using local table 1424 whether that port corresponds to a server application for which an XTP-based connection can also be accepted, and if so, it also determines whether the port is a server port for HTTP based communication. Based on this determination,

redirector 1414 either forwards a request to listen only TCP module 1416, or in addition requests either XTP module 1456 or HTTP Engine 1460 to also listen for an XTP based connection on an XTP socket. If a connection is received on an XTP-based socket, the association of the TCP socket handle known to the server application and the XTP socket handle known to the redirector is recorded in socket association table 1415.

When redirector 1414 receives any other request using a TCP socket handle, that handle is looked up in socket association table 1415 and if found, the request is forwarded to the module handling that socket (XTP module 1456 or HTTP Engine 1460), otherwise it is sent to TCP module 1416.

When HTTP Engine 1460 receives a request to send information to a client computer, HTTP Engine multiplexes the outbound data using HTTP multiplexor 1482. If the data stream corresponds to an html format Web page, the information in that page is interpreted in html parser 1483 and references to embedded data are recorded by HTTP context handler 1481. If the referenced data is available from server application 1412, HTTP context notes that this data should be prefetched from server application 1412.

When HTTP Engine 1460 receives a request to listen for a connection, that request can be satisfied in three different ways. First, if HTTP context handler 1481 has previously noted that data should be prefetched from server application 1412, a HTTP request is simulated by HTTP context handler 1481 and the listen is satisfied by this simulated request. Second, XTP module 1456 may accept a new XTP socket from a new client computer. Third, HTTP Multiplexor 1482 may satisfy the request using multiplexed communication on a XTP communication channel with a current client computer. HTTP Engine 1460

records the association of the TCP socket handle known to the server application and the source of the data that satisfied the listen request.

When HTTP Engine 1460 receives a request to read data, the read is either satisfied by HTTP context handler 1481 which simulates a HTTP request, or is handled by HTTP multiplexor 1482 depending on how the corresponding listen request was handled. Since multiple data streams may be multiplexed on a single XTP context, the read request handled by HTTP multiplexor 1482 is either satisfied by previously read data in HTTP buffer 1484, or by data read from an inbound XTP socket. Data read but that is not used to satisfy the read request is buffered in HTTP buffer 1484 until data that does satisfy the read request is found on the inbound stream.

In other embodiments, a various types of client and server communication systems can be used. Some client communication systems can include client computers communicating through a gateway computer while other client computers communicate directly to server communication systems using XTP. Some server communication systems can include one or more remote communication servers while others can use server computers that include the functionality of a remote communication server. In addition, the functionality hosted on a gateway computer in the first embodiment can be hosted on a proxy server at a POP such as proxy server 140 at POP 110b shown in Fig. 1 in which case the client communication system is hosted on computers both at the client site and at the POP. In addition, communication between a client computer and the proxy server can use a variety of protocols, including proprietary protocols that are particular to communication between certain clients and certain proxy servers.

Other embodiments of the invention use alternative methods to determine whether communication between a client computer and a server computer can use an alternative transport layer protocol, and to determine an alternative network address to which a client computer should address such communication using the alternative transport layer protocol. One approach is for the alternative network address to be a known transformation of the network address of the server application itself, for example, a different port index on the same addressed host. In this approach, a gateway computer or a specially enabled client computer tries to make a connection to a remote communication server (which may not exist) and if there is no response, assumes that the server computer is not served by a remote communication server and, instead, proceeds to establish a TCP connection.

Referring to Fig. 14, a third illustrative embodiment makes use of directory information which is distributed, in part, at server computers themselves. A client computer uses this distributed information to determine whether a server computer can accept communication using an alternative communication protocol at an alternative address. In Fig. 14, server computer 1410 is associated with a remote communication server 1420 which receives communication for server computer 1410 at an alternative address using the alternative communication protocol. Note that in related embodiments, the functionality of remote communication server 1420 and server computer 1410 could be provided by a single computer, and in such embodiments the alternative address is then on the same host as an originally requested TCP address.

Remote communication server 1420 is coupled to one or more server computers 1410 over LAN 1430 which is in

turn coupled to a client computer 1451 over Internet 1440. Client computer 1451 includes a processor 1453, non-volatile data storage 1457, such as a magnetic disk, and program storage 1454, such as a fixed or removable disk. A communication interface 1459 couples processor 1453 to Internet 1440. Server computer 1410 similarly has a processor 1413, non-volatile data storage 1416, program storage 1414 and a communication interface. Also coupled to Internet 1440 is a list server 1462, which serves a similar function to the directory servers described in the first and second embodiments. List server 1462 includes a non-volatile data storage such as a magnetic disk.

Also coupled to Internet 1440 is a domain name service (DNS) 1470 that provides a name resolution service for other computers coupled to it over the Internet. DNS 1470 can accept a host name and provide an IP address associated with that host name. List server 1462, also coupled to Internet 1440, maintains a database including associations of server computers and IP addresses of remote communication servers associated with those server computers. This database is stored on non-volatile storage 1461.

Referring to Fig. 15, when a client application 1510, such as a Web browser, executing on client computer 1451 (Fig. 14) attempts to communicate using TCP/IP with an IP address on server computer 1410 (Fig. 14), communication passes through a set of software modules also executing on client computer 1451. Fig. 15 shows an arrangement of software modules for a client computer using the Microsoft Windows95 or Windows NT operating systems.

Client application 1510 communicates with a Winsock2 module 1520 which provides an interface to communication services, including TCP/IP services needed

to communicate with remote computers. Winsock2 1520 includes interfaces to name service providers and to transport service providers. A name service provider is a software module that translates host names, in the form of character strings, into a lower-level address that are then used by transport service providers, such as a TCP service provider. A name service provider may rely on services provided by another computer, such as an Internet DNS. A transport service provider is a software module that implements transport layer communication services to allow a client application to communicate with remote applications. In Fig. 15, layered service module (LSM) 1540 provides standard transport protocol services, such as TCP-based transport services, to Winsock2 1520 according to the Microsoft Winsock2 Service Provider Interface (SPI) and therefore behaves as a transport service provider. Directory module 1550 provides IP-based name services to Winsock2 1520. Together, directory module 1550 and layered service module 1540 are part of a redirector 1530 which determines whether an alternative communication protocol can be used on a particular TCP/IP connection request from client application 1510 to a server computer, and provides an interface to transport services 1590 used to pass data to and from that server computer. Raw IP module 1595, in turn provides lower-level services to modules in transport services 1590. Transport services 1590 also provides communication services to directory module 1550. A user interface (UI) application 1535 is also coupled to directory module 1550. UI application 1535 provides an interface to a user of the client computer to access information maintained in directory module 1550, and to update that information.

Referring to Fig. 16, transport services 1590 include TCP module 1660 and XTP module 1662, which

provide transport services directly to LSM 1540. As in the previously described embodiments, XTP module 1662 is used to provide transport layer communication services for enhanced communication between a client application 1510 and a remote server application.

Continuing to refer to Fig. 16, directory module 1550 includes several submodules. A name resolution service 1650 receives name resolution requests from Winsock2 1520 and uses a database 1620 to attempt to resolve those requests. A directory manager 1610 receives requests from LSM 1540 to determine whether a remote communication server is associated with the address of a server computer. Directory manager 1610 uses database 1620 and possibly a network lookup module 1640 in responding to those requests.

Database 1620 includes an address database 1624 which includes associations of server host names, server IP addresses, and IP address of remote communication servers, if any, associated with those server hosts. Address database 1624 also includes additional information particular to specific server sites. Address database 1624 is maintained on non-volatile data storage 1457 (Fig. 14) using multiple files. In addition, database 1620 includes a memory cache 1622, stored in volatile working storage 1458 (Fig. 14), used to maintain copies of portions of the database for rapid access. Database 1620 also includes ageing information 1629 which is used to record information related to data access and is used to delete data, for example, data that was least recently accessed or has been least frequently accessed, in order to free space on working storage 1458 or non-volatile storage 1457.

Address database 1624 is stored on non-volatile storage 1457 using a separate binary file for each server computer. Referring to Fig. 17a, for a particular

server, the file contains binary fields delimited by field separators 1795. The fields in a record include the IP address of a remote communication server (SS_IP) 1770, and a name entry for the server computer (WS_NAME) 1780. The record also includes site-specific information (SITE_INFO) 1790. Each of the individual files of address database 1624 are stored in a single directory, and have file names derived from the IP address of the server computer (WS_IP). In particular, the WS_IP field is interpreted as a unsigned 32-bit number, and expressed in base 36 using the symbols 0-9 and A-Z. A common prefix, for example "SN", and a common extension, for example, ".SSF", is used for all the files. For example, the IP address "199.103.141.58" is associated with the file name "SNG8V2RR.SSF".

Referring again to Fig. 16, directory manager 1610 makes use of network lookup module 1640 to access distributed directory information that it uses to determine whether a remote communication server is associated with the address of a server computer. If, in response to a request from LSM 1540, directory manager 1610 cannot satisfy a request for remote communication server information using database 1620 alone, the directory manager issues a request to network lookup 1640. Network lookup 1640 uses path resolver 1642 and web server access 1646 modules, which in turn uses transport services 1590, to attempt to retrieve the needed information from the server computer itself. If successful, network lookup 1640 provides the retrieved remote communication server information to directory manager 1610, which then both updates database 1620 and provides a response to LSM 1540.

UI application 1535 is also coupled to directory manager 1610. UI application 1535 provides an interface for a user of the client computer to access information

in database 1620, and to update that information. A user may update information in database 1620 by explicitly providing that information to UI application 1535. For example, a user may know the address of a remote communication server associated with a particular server computer. The user may also know information needed to establish a connection to the remote communication server, such as a password, that he provides to the UI application for storage in database 1620.

Rather than explicitly providing information for database 1620, the user can also request that information be obtained from a server computer 1410 or from a list server computer 1462 (Fig. 14). UI application 1535 requests from directory manager 1610 that a particular server computer or list server, or a default list server known to directory manager 1610, be accessed to obtain information to be stored in database 1620. An explicit location on the list server may be provided by UI application 1535, or a default location known to directory manager 1610 may be used. Directory manager 1610 passes the request to path resolver 1642, which uses list server access module 1644 to retrieve the information from a list server, or which uses web server access 1646 module to retrieve the information from a server computer. The information is passed back to directory manager 1610 which uses the information to populate database 1620.

The user may also use UI application 1535 to access information in database 1620. For example, site descriptions of server computers, lists of server computers in a particular category, or other information about the servers can be displayed. This information may be used by the user to select a particular server computer.

Network lookup 1640 accesses data on a server computer, or on a list server, by retrieving a data file from that computer. Web server access 1646 uses a predetermined pathname for the data file on the server computer. The data file is known as a "site file." The file is kept in a secret directory that is not normally accessible to remote users. In this embodiment, the pathname is of the form "<WEBROOT>/<DIRNAME>/<FILENAME>.dat", for example "<WEBROOT>/.sitara/sitara.dat", where "<WEBROOT>" is the pathname of the root directory accessed by a Web server. List server access 1644 by default uses the same file name when retrieving information, although an explicit path name may be provided by the user through UI application 1535.

A site file includes information used to populate database 1620. In this embodiment, the site file is a text file. Referring to Fig. 17b, the site file has one or more lines as shown. Each line, associated with a particular server computer, contains tab delimiters 1710 separating the IP address of the server computer (WS_IP) 1730, the IP address of a remote communication server (SS_IP) 1740, which is a null entry if there is no remote communication server associated with the server computer, and a name entry for the server computer (WS_NAME) 1750. The record also includes site-specific information (SITE_INFO) 1760, and is terminated by a newline 1720. A site file, both on a server computer and on a list server, may have multiple lines corresponding to multiple associations of server computers with remote communication servers.

Operation of various software modules is illustrated in the flowcharts in Figs. 18-22. In the following description of the flowcharts, unless otherwise indicated in parentheses, elements involved in the steps

appear in Fig. 16. A request by client application 1510 to Winsock2 1520 to connect a TCP socket (communication channel) to the server computer is handled in several steps.

Referring to Fig. 18, after client application 1510 provides the name of the server computer in the form of a character string, Winsock2 1520 determines an IP address associated with this host name. Winsock2 1520 later passes that IP address to transport services that actually connect the communication channel. In determining the IP address, Winsock2 1520 first requests resolution of the server host name from name resolution module 1650 (step 1810). If name resolution module 1650 cannot find an IP address for the host (step 1812), Winsock2 1520 requests resolution of the host name from a default name service provider (step 1920) which accesses DNS 1470 (Fig. 14) to resolve the server host name. If name resolution module 1650 cannot provide the IP address (step 1812) and DNS 1470 cannot provide an IP address for the server host (step 1822), Winsock2 1520 reports a name lookup failure to client application 1510 (step 1824).

Assuming that Winsock2 1520 successfully resolves the server host name, using either name resolution service 1650 or DNS 1470, it then requests that LSM 1540 connect the socket to the IP address of the server computer (WS_IP).

Referring to Fig. 19, LSM 1540 receives a request to connect to a server at address WS_IP. LSM 1540 requests from directory manager 1610 the address of a remote communication server associated with the server computer's IP address (step 1920). Directory manager 1610 accesses database 1620 by first searching memory cache 1622 and, if no record is found, searching address database 1624. If it finds a record for the server computer, it returns the value of the SS_IP field, which

is the address of a remote communication server for the server computer. If the record was found in address database 1624, directory manger 1610 stores a copy of the record in memory cache 1622. If there is no record for the server computer in database 1620, directory manager 1610 returns a null address. If directory manager 1610 returns a null remote communication server address (step 1922), LSM 1540 attempts to connect to the server computer directly using TCP (step 1910). If directory manager 1610 returns the address of a communication server, LSM 1540 connects to the remote communication server using the appropriate enhanced communication protocol (step 1926). In particular, the enhanced protocol involves communication using XTP module 1662 in a manner presented above in the descriptions of the first and second embodiments. If LSM 1540 connects to the server computer using TCP (step 1910), it subsequently issues a request to directory manager 1610 to update its entry for that server computer (step 1915).

Referring to the flowchart of Fig. 20, when LSM 1540 requests from directory manager 1610 that it update the entry for a server computer, directory manager 1610 first determines whether a record for the server computer is stored in database 1620 (step 2010). If there is a record, directory manager 1610 does not need to perform any further update. If directory manager 1610 determines that there is no record in address database 1620 associated with the server IP address (step 2010), it attempts to obtain a site file from the server computer (step 2020). Directory manager 1610 requests the site file information from network lookup 1640 (Fig. 16) which accesses the site file from the server computer at address WS_IP, in this embodiment, using the HTTP protocol. If network lookup 1640 successfully returns the information in the site file (step 2022), directory

manager 1610 updates both memory cache 1622 and address database 1624 of database 1610 (step 2024). If the site file is not successfully retrieved (step 2022), for instance because the server computer is not associated with a remote communication server in which case the server computer returns a "file not found" message when the client computer requests the site file, directory manager 1610 updates memory cache 1622 (but not address database 1624) of database 1620 by creating a record for the server computer, and storing a null in the field for the remote communication server address associated with the server address (step 2026). Note that memory cache 1622 is deleted when client application 1510 exits, and therefore a record which indicates that a server computer is not associated with a remote communication server persists only as long as the application that accessed the server computer is active.

A fourth embodiment extends the functionality of the third embodiment, described above, to include loadable software modules for use in communication with particular remote communication servers. Referring back to Fig. 17, the SITE_INFO field of a record of database 1620 can now include a reference to a loadable software module that is required for communicating with the remote communication server at address SS_IP. A reference includes a URL of the software module. Directory manager 1610 is responsible for retrieving and keeping the software module up to date. LSM 1540 is responsible for dynamically loading the module, and for branching to available entry points in the module at appropriate times while communicating with the remote communication server. A loadable module is kept up to date using a trigger mechanism. Specifically, for each loadable module that has previously been retrieved, directory manager 1610 maintains records the latest date the module was

retrieved or that it checked to determine whether a newer version was available. Each loadable module is also associated with a maximum interval that the module can be used before directory manager 1610 must check that a newer version is available. If more than the maximum interval for a module has elapsed since the module was loaded or a check was made, directory manager 1610 checks whether a newer version is available, and if one is available, the newer version is retrieved by directory manager 1610 and loaded by LSM 1540 without the necessity of user intervention.

Referring to Fig. 21, after directory manager 1610 retrieves a record from database 1610, or from a newly retrieved site file, (step 2110), directory manager 1610 determines from the SITE_INFO field whether a loadable module is required for communication with the indicated remote communication server (step 2112). If a loadable modules is not needed, the remote communication server IP address is returned to LSM 1540 as in the third embodiment. If a loadable modules is required (step 2112), then directory manager 1610 determines whether it is currently storing a local copy of the required module (step 2114). If not, the directory manager retrieves the module (step 2120) and stores it in a local file. The IP address of the remote communication server and the pathname of a local copy of the retrieved module are returned to LSM 1540. If the module has been previously retrieved (step 2114), then it may have already expired. Expiration is checked (step 2116), and if it the module has not expired, then the IP address of the communication server and the local pathname of the loadable module are passed to LSM 1540. If, on the other hand, the module has expired (step 2116), directory manager 1610 checks whether a newer version is available by accessing the date of the version at the referenced location over the

Internet (step 2118). If a newer version is available, that version is retrieved and stored in a local file (step 2120). The IP address of the communication server and the local pathname of the loadable software module are then passed to LSM 1540.

Referring to Fig. 22, LSM 1540 includes basic layered services 2220 and possibly one or more loaded modules 2230. In this embodiment, which is implemented using the Windows95 or Windows NT operating system, a loadable software module is a Dynamically Linked Library (DLL). When LSM 1540 receives a local pathname of a loadable module, that is, the pathname of a DLL, LSM 1540 loads the DLL. The DLL uses an agreed-upon API, and includes a module descriptor which allows basic layered services 2220 to determine what routines that DLL provides. The module descriptor is also used to allow proper chaining of modules if more than one routine needs to be used in sequence. For example, a decompression routine and a decryption routine both may need to be used on an inbound data stream, and the order of applying those routines is significant to their proper operation. Using the module descriptor, LSM 1540 determines under what circumstances a routine in the loadable module should be invoked.

Use of such a loadable module can be understood by an example in which the module is used for decompression of data flowing from the communication server to the client computer. In this case, loadable module 2230 associated with the communication server includes an entry point used to filter an incoming data stream. Each input buffer read by basic layered services 2220 from transport services 1590 is passed to the inbound filtering routine in loadable modules 2230. The result of applying the routine is then passed to Winsock2 1520 and then to client application 1510. Entry points can

also be associated with and invoked by LSM 1540 in a variety of situations, for instance, when connecting, closing, reading from, and writing to a socket.

Referring still to Fig. 22, loadable modules communicate with basic layered services 2220. In the example introduced above in which a loadable module is used for decompression of data, compressed data received by the client computer is passed from basic layered services 2220 to loadable module 2230, where it is decompressed, and decompressed data is passed back from loadable module 2230 to basic layered services 2220. The decompressed data is then sent to client application 1510. Similar data flow is used for a loadable module that provides decryption routines.

A loadable module may also communicate directly with transport services 1590. For example, a loadable module may multiplex multiple socket connections over a single TCP connection. In this case, basic layered services 2220 would provide outbound data to loadable module 2230, and loadable module 2230 would provide that data, after multiplexing it with data from other connections, to TCP module 1660.

A loadable module may also essentially implement a transport service, for instance, if a proprietary transport layer protocol is used with a particular server computer. Such a protocol may be used, for example, to provide streaming audio data in an efficient manner. In this case, a loadable module 2230 communicates directly with Raw IP 1595.

A loadable module may also communicate with an application interface 2210 which is a separate application from client application 1510 which initiated the connection. In this way, a customized client server application may be initiated. For example, a customized user interface, particular to a server computer, can be

launched by a loadable module. The loadable module then provides an interface for application interface 2210 and communication services needed by application interface.

The description of the third and fourth embodiments referred to a single database entry in database 1620 being associated with a server computer. Several remote communication server computers may in fact be associated with a particular server computer. In this case, the file in address database 1622 associated with that server computer will have multiple records of the format shown in Fig. 17b. In the third embodiment, directory manager 1610 implements a selection of which communication server address to provide to LSM 1540. For instance, directory manager 1610 may implement a round-robin scheme in which each request by LSM 1540 to connect to the server computer results in a different remote communication server being used than in the previous connection. Directory manager 1610 may also implement another selection method, for example based on other information recorded in the site information field of the database. For instance, the site information field may indicate that one remote communication server is appropriate for clients with a high data rate connection to the remote communication server, while another communication server is appropriate for clients with slow or high-latency connections.

Also in the description of the third and fourth embodiments, the system was described in terms of a local computer establishing an enhanced communication path to a server computer. In the first two embodiments, an enhanced communication path used an enhanced transport layer protocol, that is the XTP protocol, or multiplexing of socket connections from a client application using a single TCP or XTP. In the third and fourth embodiments, communication between a local computer and a remote

server may be enhanced in a wide variety of ways, not necessarily involving transport layer communication protocols. For example, application layer enhancements to the communication between the client computer and the remote computer may be used. For instance, if a server computer supports a catalog shopping service, communication passing between the local computer and the server computer may involve relatively low level data requests, for example for product descriptions, prices, etc. The user, however, may be presented a full-featured user interface within a standard Web browser. Software in the layered service module, or in a loadable module, provides the interface between the Web browser and the low level communication passed between the client computer and the server computer. Other enhanced communication approached implemented at various layers, from the data link layer to the application layer, may be used in the described, or related, embodiments.

In the fourth embodiment, if multiple remote communication servers are available for a server computer, the entire list is provided to LSM 1540. A loadable module is then be used to select the appropriate communication server for each connection.

Also in the fourth embodiment, other forms of loadable modules may be used. For instance, rather than DLLs, Java applets may be used.

In the third and fourth embodiments, database 1620 is implemented using text files stored on the client computer's file system to store database records and loadable software modules. An object database may alternatively be used. For instance, each record may be stored as a binary file including a tag and length for each field, multiple records may be stored in a single file, or relational tables may be used. Furthermore, if loadable software modules are used, database 1620 may be

used to store the modules themselves, rather than using the local computer's file system.

Also in the third and fourth embodiments, use of the cooperating name resolution service 1650 and layered service module 1540 allows a client application to use host names that would not be resolved by an Internet DNS. For instance, referring to Fig. 15, a "friendly" name can be passed from client application 1510 to Winsock2 1520. The friendly name is passed to directory module 1550 which returns a data structure which includes the address in the form needed by the appropriate transport services.

An alternative approach to determining whether a server computer can be accessed through a remote communication server involves a client computer (or proxy computer) initially retrieving information from a server computer using TCP. Then, based on application layer data transferred over the TCP connection, the client computer determines the address of a remote communication server. In the case of http-based data connections, the address of the remote communication server is provided as part of the response to a HTTP request. The server address can be embedded in the stored files on the server computer, thereby being provided to the client computer as part of a normal retrieval of data by a standard Web server application. Alternatively, the remote communication server address can be added by the Web server application prior to sending a response to the client. If the client receives a remote communication server address in the data, it updates its local tables and uses the remote communication server for subsequent communication with the server computer.

Referring to Fig. 23, in an alternative arrangement, remote communication servers 2340 can be used to balance load on a set of server computer in conjunction with a network address translator (NAT) 2310.

NAT 2310 is connected between the Internet and a LAN 2350. Network addresses of computers on LAN 2350 are not necessarily known to computers, such as client computers, on the Internet. In order to communicate with a computer on LAN 2350, a computer on the Internet directs communication to a network address to which NAT 2310 responds and NAT 2310 forwards communication between the computer on the Internet and the computer on LAN 2350. Internally, NAT 2310 maintains an address association table 2312 which maps Internet network addresses to local network addresses which are valid on LAN 2350. In the simplest case, the table has a one-to-one correspondence between Internet network addresses and local network addresses. Address association table 2312 can also have multiple local network addresses associated with a single Internet network address. In this case, NAT 2310 chooses one of the multiple local network addresses to match to the Internet network address when it receives communication from a remote computer. For example, NAT 2310 can use a round-robin selection approach to choose a particular local network address. NAT 2310 can therefore balance the load of requests to communicate with a single Internet network address by passing different requests to different computers on LAN 2350. Once NAT 2350 has connected a remote computer on the Internet to a particular local computer on LAN 2350, it connects subsequent requests from that remote computer to the same local computer on LAN 2350 until a minimum time has elapsed since the remote computer has closed all its communication paths to the local computer.

Referring still to Fig. 23, LAN 2350 couples two or more remote communication servers 2340 and two or more server computers 2330 to NAT 2310. Server computers 2330 share a common network address and remote communication servers 2340 share anther network address. Requests from

a client system to communicate directly with a server computer 2330 are handled by NAT 2310 in the manner described above. Requests from a client system to communicate with a remote communication server are also handled by NAT 2340. NAT 2340 handles these requests by choosing one of the remote communication servers 2340 and passing communication between the client system and the remote communication server. This communication includes the XTP communication over which the client system sends requests to communicate with server computers.

When a remote communication server 2340 receives a requests to communicate with a server computer, the request identifies the server computer using a network address that would be valid from the Internet, that is, using an address that would be translated by NAT 2340 if the client system were to try to contact the server computer directly. Therefore, remote communication server maintains a server association table 2342 containing the Internet network address for the server computers associated with their local network addresses. When remote communication server 2340 receives a requests to communicate with a server with a particular Internet network address, the remote communication server accesses its server association table to determine whether it provides communication services for that server computer and, if so, to determine a local network addresses correspond to that server. Server association table 2342 also includes a record of the particular server computers matched to requests from particular client systems in order that the same server computer can be uses for a series of requests to communicate.

In the arrangement shown in Fig. 23, address translation therefore occurs at two levels. First, NAT 2310 translates an Internet network address for a remote communication server into one of the local network

addresses of remote communication servers. Then, when a client requests to communicate with a server computer, the remote communication server translates the Internet network address of the requested server computer into one of the local addresses of the server computers on the LAN.

Other combinations of elements described in the above embodiments may alternatively be used. For instance, the first embodiment uses a client computer which accesses the Internet through a gateway computer. Other described embodiments combine the functionality of the client computer and the gateway computer into a single computer. In embodiments that combine the functions, separate computers could equivalently be used. Similarly, the functions of a remote communication server and a server computer may be hosted on separate computers, or hosted on a single computer. If their function is hosted on a single computer, the communication passing between the remote communication server and the server computer does not have to pass over a data network.

In the described embodiments, information related to a server computer, and in particular information associating a server computer with a remote communication server, is stored in some combination of a directory server, a list server, or on the server computer itself. Other storage locations for the information may equivalently be used. For example, information related to loadable modules may be stored on a centralized directory server. Also, the function of a DNS may be extended so that, in addition to providing a translation of a host name to its associated IP address, additional information, including an address of a remote communication server, may be provided. In the case that additional information is stored on the DNS computers and

retrieved along with the IP address of a server computer, there is no explicit request for the information. The additional information is provided in response to the request for the server computer's IP address. The address of a remote communication server associated with a server computer may also be provided without an explicit request in other situations. For example, the information may be provided in the header of information retrieved from a server computer. If the server computer is a Web server, the address of a remote communication server associated with it may be provided in the header of HTML documents retrieved from the server. The client computer is then responsible for detecting and extracting the information from those HTML documents as they are passed to an application.

If the function of a remote communication server and a server computer are combined on a single computer, the host portion of the address of a server computer and the remote communication server would, in general, be the same. Communication can be delivered to the system on the single computer by using different port indices for the function of the server computer and the function of the remote communication server. Alternatively, or in addition to, using a different port index, communication from the client computer to the server computer can use a different protocol index than communication from the client computer to the remote communication server. Communication can be routed to the appropriate system on the single computer based on the protocol index.

In the above embodiments, there may be some remote communication servers that do not advertise their address in a directory server, in a list server, or in a site file. For example, access to a particular remote communication server may be limited to particular subscribers of a service while others use the standard

TCP interface. In such a case, a manual entry of the remote communication server information may be possible. Also, in the cases that a directory server is used, requests from a gateway computer may be batched and communication between a gateway computer and a directory server can occur less frequently thereby reducing the load on the directory server.

In other related embodiments, multiple alternative protocols can be supported. Instead of passing all non-TCP communication through an XTP module, a matrix switch feeding multiple protocol modules can be used. The redirector then also determines which protocol module to use. Software modules, including the "hook," the redirector, or the HTTP Engine, may be incorporated into a layered Winsock protocol under the Windows 95 or Windows NT operating system.

Additional application protocol spoofing can be performed. For example, an FTP module can be inserted between a redirector and the XTP module to spoof FTP communication in the same way that a HTTP Engine is used to spoof the HTTP application protocol. Various techniques may be used to initialize a socket using an alternative protocol. Rather than recording requests in a tracing buffer, the socket may be initialized directly into a desired state.

In addition to adding alternative transport protocols which use IP, replacing the IP layer as well may be feasible in some situations. For example, if IP is layered on an ATM network, both TCP and IP can be bypassed in a similar arrangement to that used to bypass TCP in the described embodiments.

Embodiments of this invention can make use of multiple TCP segments rather than using XTP or some other alternative transport protocol. In such embodiments, application protocol spoofing, multiplexing, and server

site aggregation (service of multiple server computers using a single remote communication server) can be used over TCP channels. In addition, the parameters of the TCP connections on different segments may be different resulting in improved end-to-end characteristics.

Other embodiments may address channel characteristics other than throughput and latency, or address throughput and latency using different types of communication techniques. For example, achieving low latency over high capacity but high delay channels may be best be achieved by using an alternative transport protocol that makes use of forward error correction rather than error detection and retransmission. In addition, the alternative protocols can be used to control a quality of service on certain data streams while still allowing the application to use TCP without modification.

Software used to implement various components of the invention may be stored on a variety of computer readable media, including fixed or removable magnetic or optical disks. Alternatively, it may be stored remotely from the computer on which the modules execute, and accessed using a data network.

It is to be understood that while the invention has been described in conjunction with the detailed description thereof, the foregoing description is intended to illustrate and not limit the scope of the invention, which is defined by the scope of the appended claims. Other aspects, advantages, and modifications are within the scope of the following claims.

What is claimed is:

1. A communication system comprising a client communication system coupled to a data network for communicating with a plurality of server communication systems each configured to communicate with the client communication system using at least one of a plurality of transport layer protocols, wherein the client communication system includes:

a transport layer module implementing the plurality of transport layer protocols for communicating with the server communication systems; and

a layered communication module coupled to the transport layer module and including a protocol selector for receiving a request to communicate with a requested one of the plurality of server communication systems and, using the request to communicate, choosing one the plurality of transport layer protocols for communication with the requested server system.

2. The communication system of claim 1 wherein the client communication system further includes an application coupled to the layered communication module.

3. The communication system of claim 2 wherein the client communication system includes a client computer programmed to implement the transport layer module, the layered communication module, and the application.

4. The communication system of claim 2 wherein the application includes a user interface module for accepting from a user requests to communicate with server communication systems and for presenting to the user information sent from the server communication system.

5. The communication system of claim 2 wherein the application includes a server module coupled to a plurality of client applications over a local data network for accepting requests from the client applications to communicate with server communication systems and for passing information between the server communication systems and the client modules.

6. The communication system of claim 5 wherein the client communication system further includes a proxy computer coupled to the local data network and programmed to implement the transport layer module, the layered communication module, and the application.

7. The communication system of claim 5 wherein the client communication system further includes a client computer coupled to the local data network and programmed to implement at least one of the client applications.

8. The system of claim 7 wherein the client communication system further includes a second client computer coupled to the proxy computer over the local data network and programmed to implement another of the client applications.

9. The communication system of claim 1 wherein the client communication system further includes an application including the layered communication module.

10. The communication system of claim 1 wherein the client communication system further includes a server table coupled to the protocol selector of the layered communication module for holding information related to the transport layer protocols with which zero or more of

the server communication systems are configured to communicate.

11. The communication system of claim 10 wherein the server table includes a first table for holding information associating identifications of server communication systems and information related to one or more of the transport layer protocols that the client communication system can use to communicate with each of the identified server communication systems.

12. The communication system of claim 11 wherein the information related to the transport layer protocols includes an address of a communication server computer in the server communication system coupled to the data network for communicating with the identified server communication system.

13. The communication system of claim 11 wherein the server table further includes a second table for holding information associating identifications of server communication systems and identifying one or more transport layer protocols with which the identified server communication systems are not configured to communicate.

14. The communication system of claim 10 wherein the client communication system further includes a table interface module for providing a user interface for viewing and modifying information in the server table.

15. The communication system of claim 10 wherein the information related to the transport layer protocols includes information related to an expiration time after

which some of the information related to the transport layer protocols is not longer valid.

16. The communication system of claim 1 wherein the transport layer module includes a rate control module for negotiating a maximum rate of data transmission with server communication systems.

17. The communication system of claim 1 wherein the layered communication module includes a data compression and decompression module for compressing and decompressing data communicated using at least one of the transport layer protocols.

18. The communication system of claim 1 wherein the layered communication module includes a security module for encrypting and decrypting data communicated using at least one of the transport layer protocols.

19. The system of claim 1 wherein at least one of the transport layer protocols supports selective retransmission and at least one other of the transport layer protocols does not support selective retransmission.

20. The system of claim 1 wherein the layered communication module further includes a multiplexor and a demultiplexor for combining a number of data streams associated with separate requests to communicate with server communication systems into a smaller number of transport layer data streams for communicating with the server communication systems.

21. The communication system of claim 1 wherein the client communication system further includes a

directory service module coupled to the protocol selector of the layered communication module for accessing over the data network information related to the transport layer protocols with which server communication systems are configured to communicate.

22. A communication system comprising a server communication system coupled over a data communication network to a plurality of client communication systems, the server communication system including:

a transport layer module for communicating with the client communication systems and one or more server applications; and

a communication application coupled to the transport layer module for maintaining a transport layer communication stream with each of a number of client communication systems, for accepting requests over the communication streams from client communication systems to communicate with the one or more server applications, and for passing information between the client communication systems and the server applications over the communication streams.

23. The communication system of claim 22 wherein the server communication system further includes a communication server computer coupled to the data communication network and programmed to implement the transport layer module and the communication application.

24. The communication system of claim 23 wherein the communication server computer is further programmed to implement at least one of the server applications.

25. The communication system of claim 23 wherein the server communication system further includes an

application server computer programmed to implement at least one of the server applications and a local data network coupling the communication server computer and the application server computer.

26. The communication system of claim 22 wherein the transport layer module implements a plurality of transport layer protocols, and wherein the communication application is configured to communicate with the server applications using a first of the transport layer protocols and to communicate with client communication systems using a second of the transport layer protocols.

27. The communication system of claim 26 wherein the second transport layer protocol supports selective retransmission and the first transport layer protocol does not support selective retransmission.

28. The communication system of claim 22 wherein the communication application is configured to limit accepted requests from client communication systems to communicate with server applications based on a total allocated communication rate of accepted requests.

29. The communication system of claim 22 wherein the server communication system further includes a server table identifying server applications for which the communication application is configured to pass information to a client communication system.

30. The communication system of claim 29 wherein the server communication system further includes a module for accepting information to update the server table.

31. The communication system of claim 22 wherein the communication application is configured to pass information between more than one server application and a client communication system over a single transport layer communication stream.

32. The communication system of claim 22 wherein the server communication system further includes a server table used to identify server computers with which client computers are permitted to communicate.

33. The communication system of claim 22 wherein the server communication system further includes an address translation table for associating network addresses provided by client communication systems as identifiers of server applications with local network addresses used for communicating between the communication application and the server applications.

34. The communication system of claim 33 wherein the address translation table is configured to associate more than one local network address for each network address provided by a client communication system, and the server communication system further includes a server selection module for selecting one of the local addresses in response to a request to communicate from a client communication system.

35. The communication system of claim 34 wherein the server selection module includes a table for storing an association between a client communication system and a selected local network address.

36. The communication system of claim 22 wherein the transport layer module includes a rate control module

configured to control a total rate of communication to each client communication system.

37. The communication system of claim 22 wherein the communication application includes a compression module for compressing information passing from a server application to a client communication system.

38. The communication system of claim 22 wherein the communication application includes an encryption module for encrypting information passing from a server application to a client communication system.

39. A communication system comprising a server communication system coupled over a data network to a plurality of client communication systems, the server communication system including:

a transport layer module for communicating with the client communication systems using a plurality of transport layer protocols;

a server application; and

a layered communication module coupled between the transport layer module and the server application for receiving requests from the server application to accept communication from the client communication systems, for maintaining a transport layer communication stream with each of a number of the client communication systems, for accepting requests over the communication streams from client communication systems to communicate the server application, and for passing information between the client communication systems and the server application over the communication streams.

40. The communication system of claim 39 wherein the server communication system further includes a server

computer coupled to the data network and programmed to implement the transport layer module, the server application, and the layered communication module.

41. A method for communicating between a client communication system and a plurality of server communication systems over a data communication network comprising:

accepting a request to communicate including receiving an identification of one of the server communication systems;

using the identification of the server communication system, selecting one of a plurality of transport layer protocols for communicating with the server communication system; and

communicating with the server communication system over the data communication network using the selected transport layer protocol.

42. The method of claim 41 wherein selecting the transport layer protocol includes determining a set of one or more transport layer protocols for which the server communication system is configured to communicate.

43. The method of claim 42 wherein determining the set of protocols includes accessing a server table and retrieving information related to the server communication system.

44. The method of claim 43 wherein determining the set of protocols further includes updating the server table based on data received from the server communication system.

45. The method of claim 42 wherein determining the set of protocols includes retrieving information related to the server communication system from a directory computer over the data communication network, and wherein the address of the directory computer is related to the identification of the server communication system.

46. The method of claim 45 wherein the address of the directory computer is the same as the address of a server computer identified in the request to communicate.

47. The method of claim 41 wherein accepting the request to communicate includes accepting a request to communicate with the server communication system using a requested transport layer protocol for which the server communication system is configured to communicate, and the selected transport layer protocol is different from the requested transport layer protocol.

48. The method of claim 41 wherein accepting the request to communicate with the server communication system includes accepting a request to communicate with a server computer at a first network address over the data communication network, and wherein the method further includes selecting a second network address for communicating with the server communication system, and wherein communicating with the server communication system includes communicating with a computer at the second network address.

49. The method of claim 48 wherein the second network address is different from the first network address.

50. The method of claim 41 further comprising:

accepting a request for information through a user interface;

providing a request to communicate with a server communication system based on the request for information; and

providing the requested information through a user interface.

51. The method of claim 41 wherein accepting the request to communicate further includes accepting a request to communicate from a client application over a local data network.

52. A method for communicating between a server communication system and a plurality of client communication systems over a data communication network comprising:

establishing a first transport layer communication path between the server communication system and one of the client communication systems using a first transport layer protocol;

receiving over the transport layer communication path a request from the client communication system to communicate with a first server application;

establishing a second communication path to the first server application using a second transport layer protocol; and

passing communication between the client communication system and the first server application over the first transport layer communication path and over the second transport layer communication path.

53. The method of claim 52 wherein passing communication includes selectively retransmitting data

not correctly received by the client communication system.

54. The method of claim 52 wherein passing communication includes limiting a total rate of communication with the client communication system.

55. The method of claim 52 further comprising:

receiving over the first transport layer communication path a second request to communicate with a second server application;

establishing a third transport layer communication path to the second server application using the second transport layer protocol; and

passing communication between the client communication system and the second server application over the first transport layer communication path and over the third transport layer communication path.

56. The method of claim 55 wherein the first server application and the second server application are the same and the second transport layer communication path is separate from the third transport layer communication path.

57. The method of claim 55 further comprising accessing an address translation table and determining a local network address of the first server application.

58. The method of claim 55 wherein passing communication includes limiting a total rate of communication with the client communication system.

59. A method for communicating between a server communication system and a plurality of client

communication systems over a data communication network comprising:

establishing a first transport layer communication path between the server communication system and one of the client communication systems using a first transport layer protocol;

receiving over the transport layer communication path a request from the client communication system to communicate with a first server application;

determining whether to accept the request; and

if the request is accepted establishing a second communication path to the first server application using a second transport layer protocol, and passing communication between the client communication system and the first server application over the first transport layer communication path and over the second transport layer communication path.

60. The method of claim 59 wherein determining whether to accept the request includes accessing a server table identifying server applications for which communication should be passed.

61. The method of claim 59 wherein determining whether to accept the request includes determining whether accepting the request would exceed a total communication capacity.

62. Software stored on a computer readable medium for causing a computer to perform the functions of:

accepting a request to communicate including receiving an identification of one a plurality of server communication systems;

using the identification of the server communication system, selecting one of a plurality of

transport layer protocols for communicating with the server communication system; and

communicating with the server communication system over the data communication network using the selected transport layer protocol.

63. Software stored on a computer readable medium for causing a computer to perform the functions of:

establishing a first transport layer communication path between a server communication system and one of a plurality of client communication systems using a first transport layer protocol;

receiving over the transport layer communication path a request from the client communication system to communicate with a first server application;

establishing a second communication path to the first server application using a second transport layer protocol; and

passing communication between the client communication system and the first server application over the first transport layer communication path and over the second transport layer communication path.

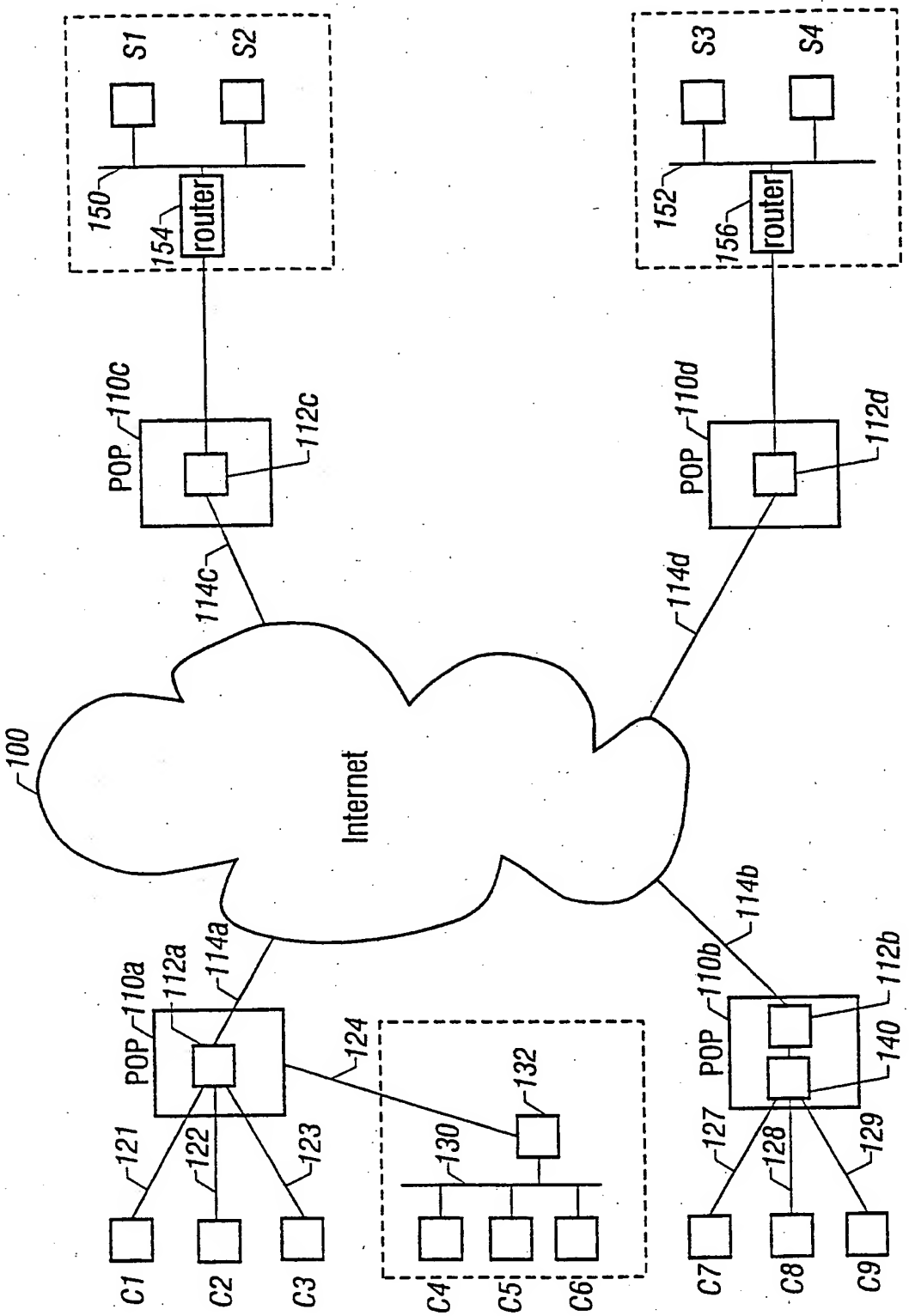


FIG. 1
(Prior Art)

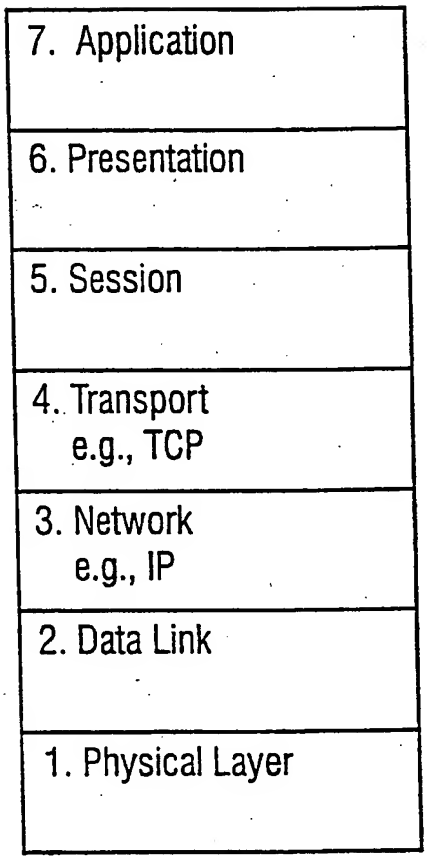


FIG. 2
(Prior Art)

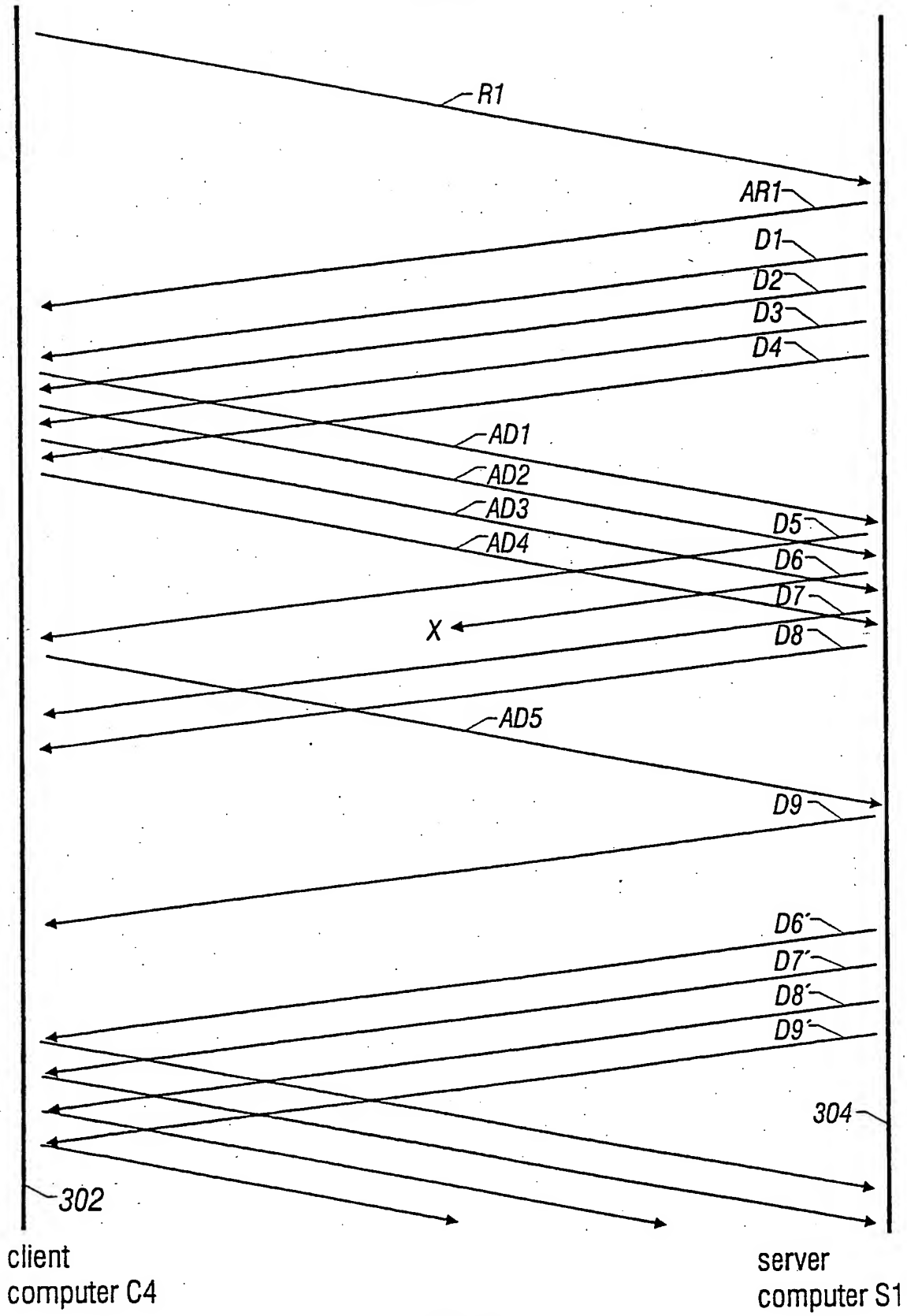

R1
AR1
D1
D2
D3
D4
AD1
AD2
AD3
AD4
D5
D6
D7
X
D8
AD5
D9
D6'
D7'
D8'
D9'
304
302
client
computer C4
server
computer S1

FIG. 3

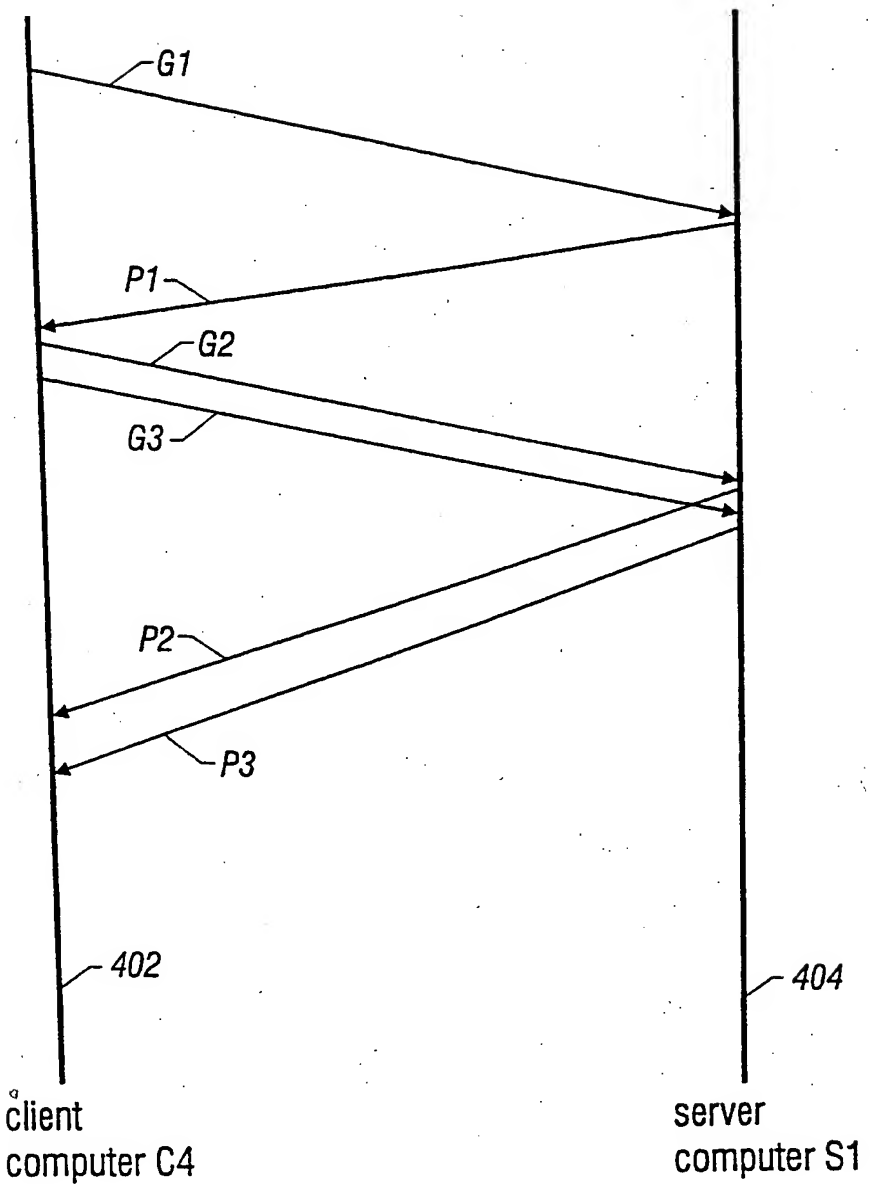


FIG. 4
(Prior Art)

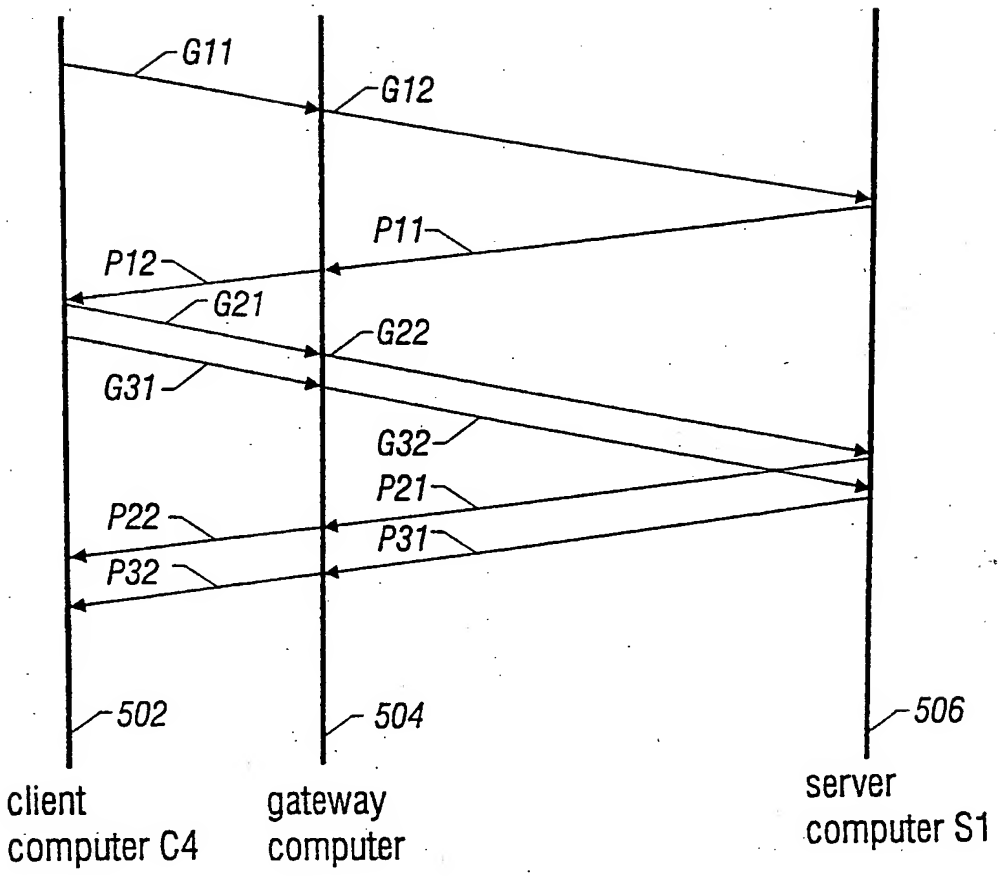


FIG. 5
(Prior Art)

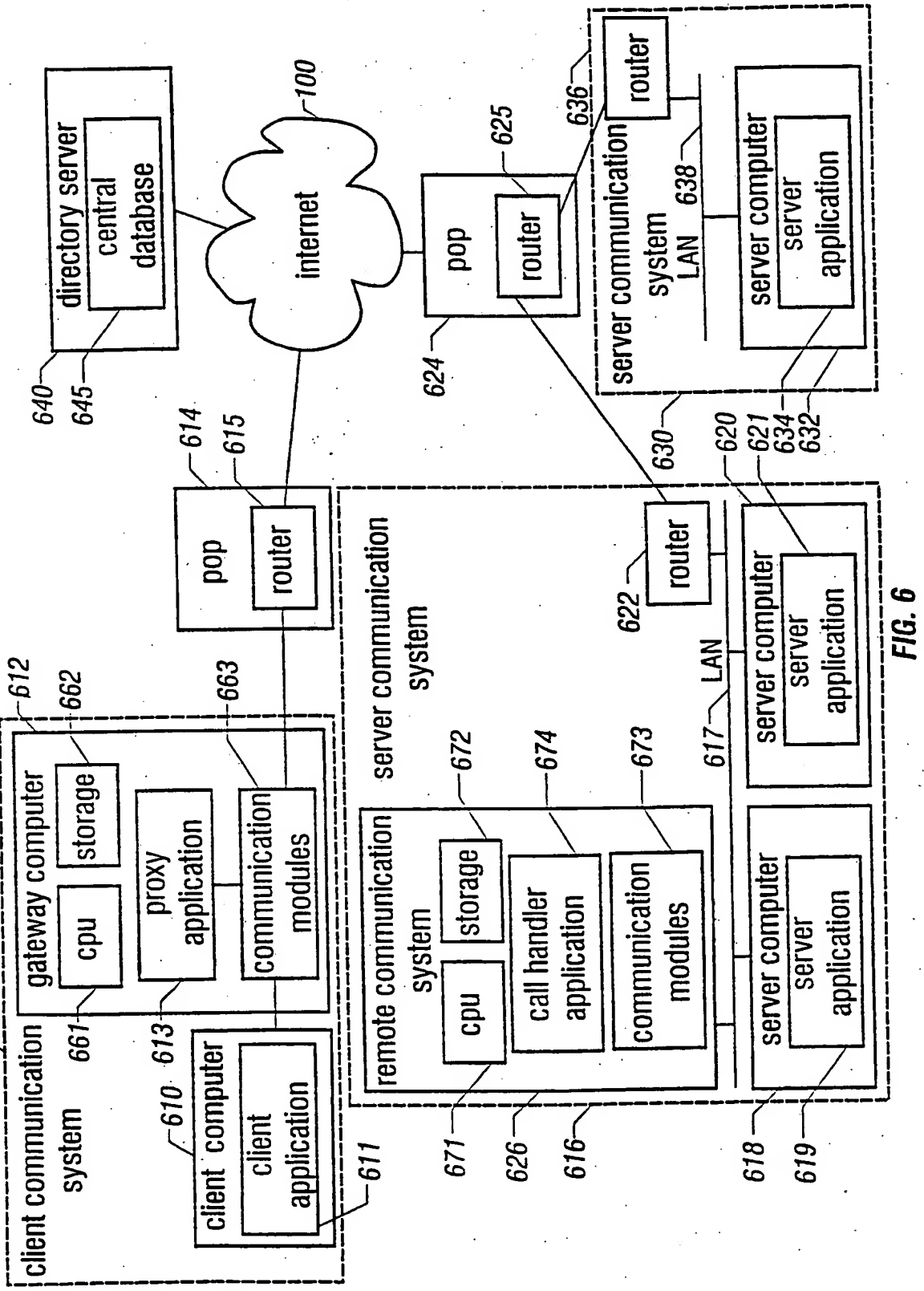

directory server
central database
640
645
internet
100
pop
router
614
615
625
636
server communication system
LAN
638
server computer
server application
624
630
620
621
634
632
622
router
server computer
server application
LAN
617
client communication system
606
gateway computer
cpu
storage
661
662
proxy application
613
communication modules
663
612
client computer
client application
610
611
remote communication system
cpu
storage
672
call handler application
674
communication modules
673
671
626
616
server computer
server application
618
619

FIG. 6

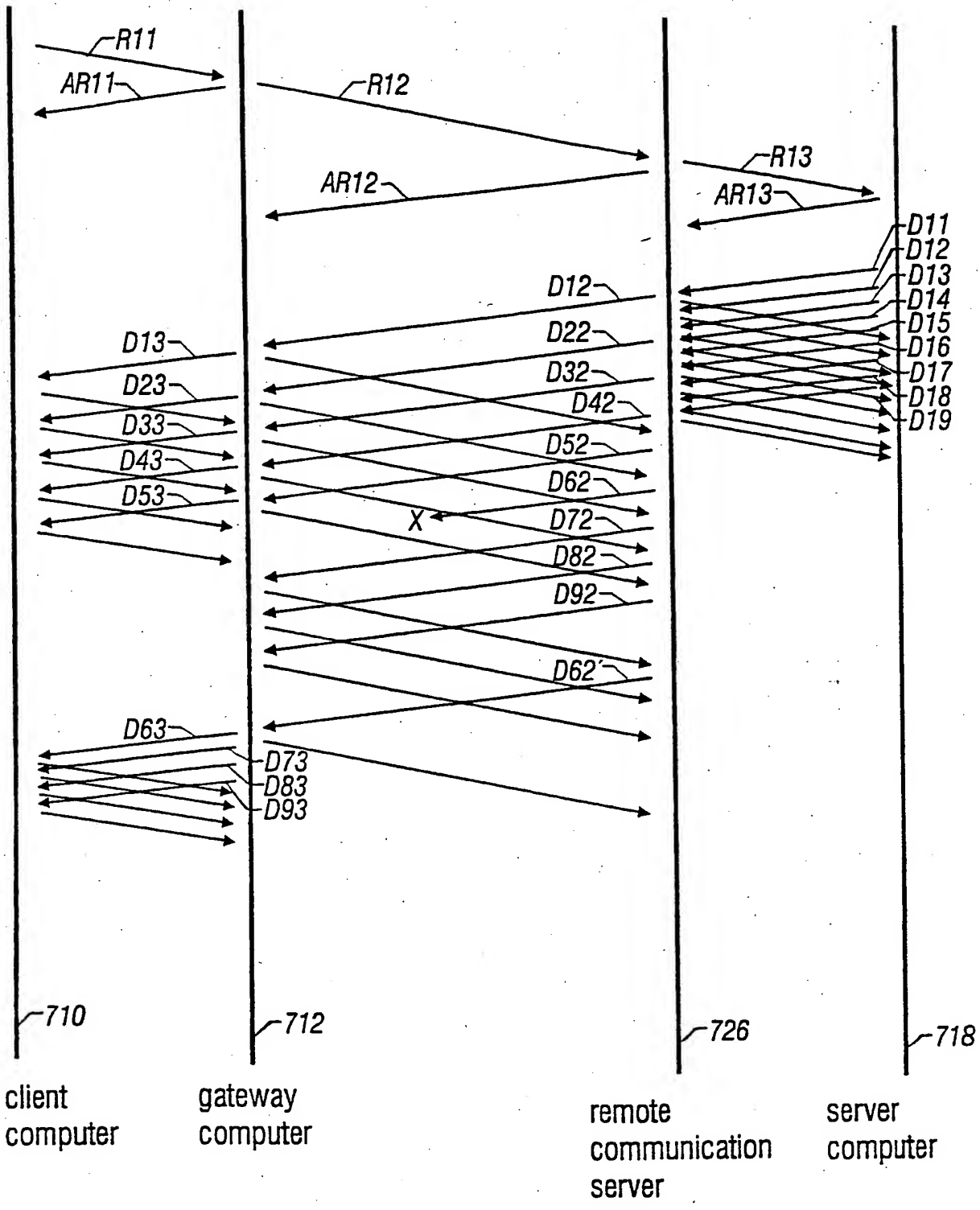

R11
AR11
R12
AR12
R13
AR13
D11
D12
D13
D14
D15
D16
D17
D18
D19
D12
D22
D32
D42
D52
D62
D72
D82
D92
D62'
X
D13
D23
D33
D43
D53
D63
D73
D83
D93
710
712
726
718
client computer
gateway computer
remote communication server
server computer

FIG. 7

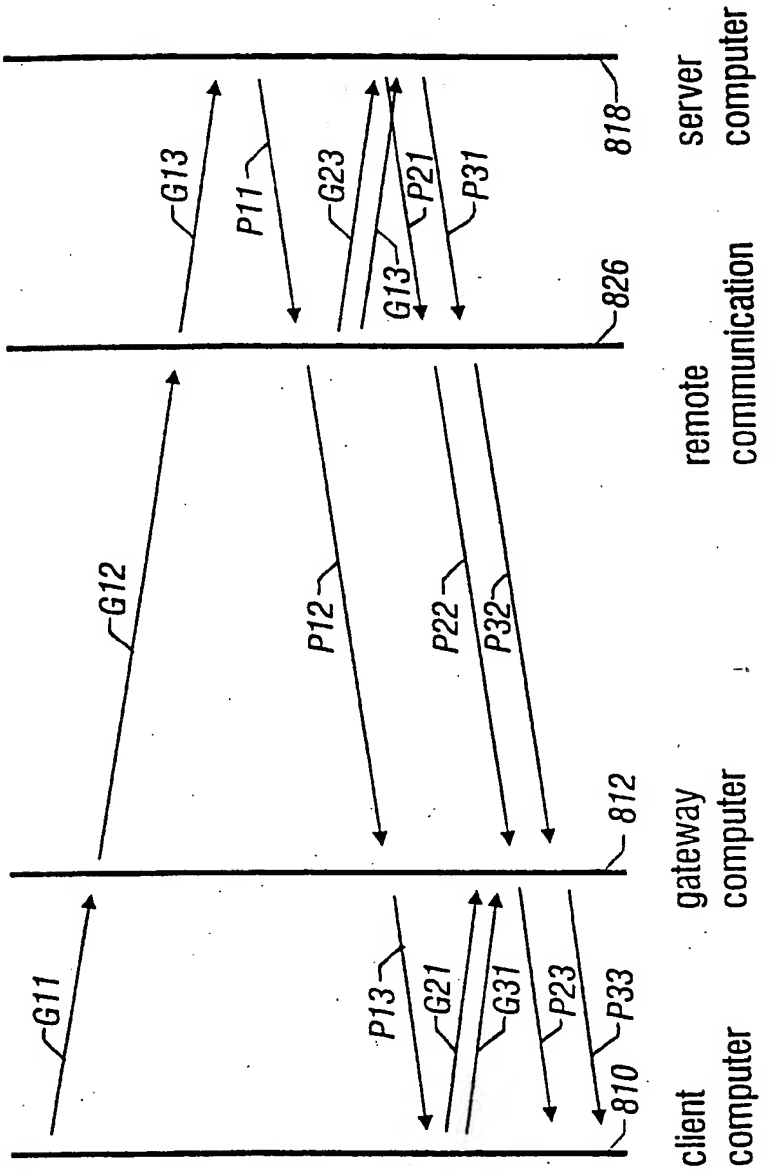

G13
P11
G23
P21
P31
G13
818
826
server computer
remote communication server
G12
P12
P22
P32
812
gateway computer
G11
P13
G21
G31
P23
P33
810
client computer

FIG. 8

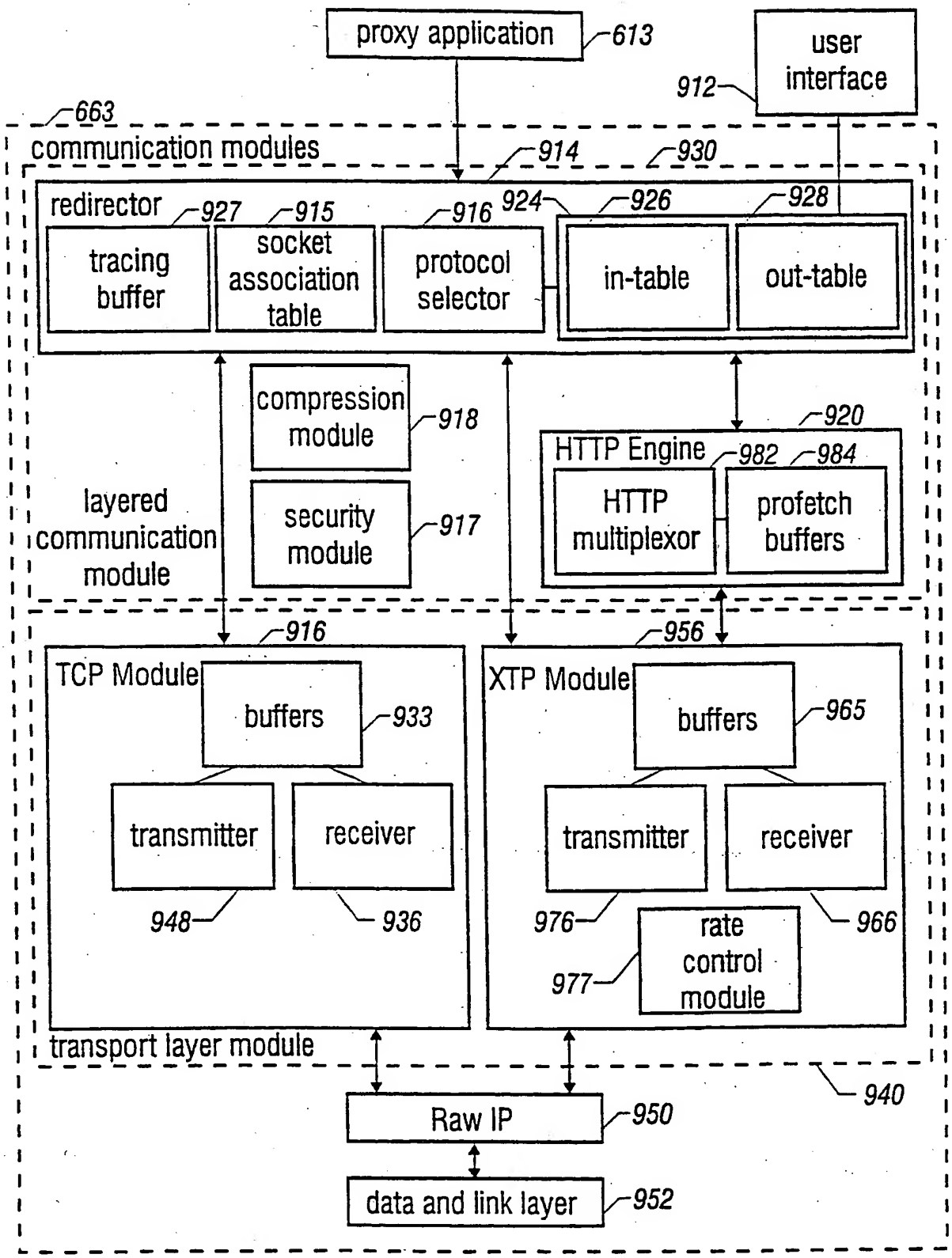

proxy application
613
user interface
912
663
communication modules
914
930
redirector
927
915
916
924
926
928
tracing buffer
socket association table
protocol selector
in-table
out-table
compression module
918
920
HTTP Engine
982
984
HTTP multiplexor
profetch buffers
layered communication module
security module
917
916
956
TCP Module
buffers
933
XTP Module
buffers
965
transmitter
receiver
transmitter
receiver
948
936
976
966
rate control module
977
transport layer module
940
Raw IP
950
data and link layer
952

FIG. 9

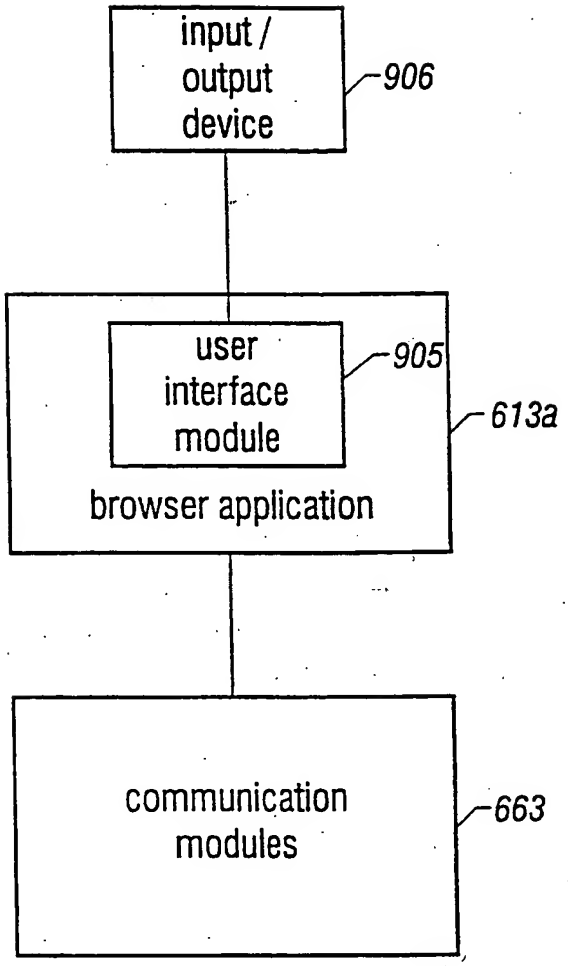

input / output device
906
user interface module
905
613a
browser application
communication modules
663

FIG. 9A

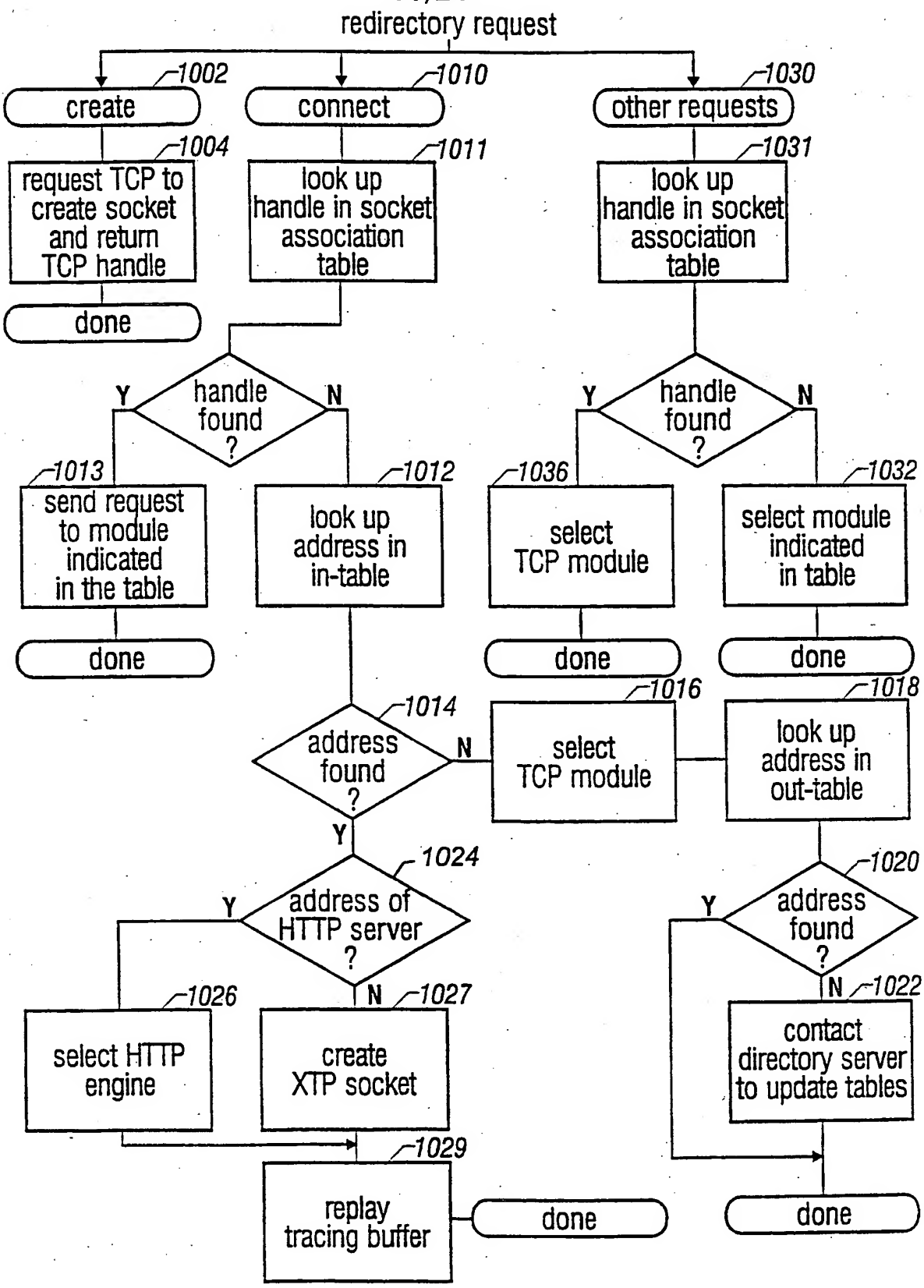

redirectory request
create
1002
request TCP to create socket and return TCP handle
1004
done
connect
1010
look up handle in socket association table
1011
handle found ?
Y
N
send request to module indicated in the table
1013
done
look up address in in-table
1012
address found ?
1014
Y
N
address of HTTP server ?
1024
Y
N
select HTTP engine
1026
create XTP socket
1027
replay tracing buffer
1029
done
other requests
1030
look up handle in socket association table
1031
handle found ?
Y
N
select TCP module
1036
done
select module indicated in table
1032
done
select TCP module
1016
look up address in out-table
1018
address found ?
1020
Y
N
contact directory server to update tables
1022
done

FIG. 10

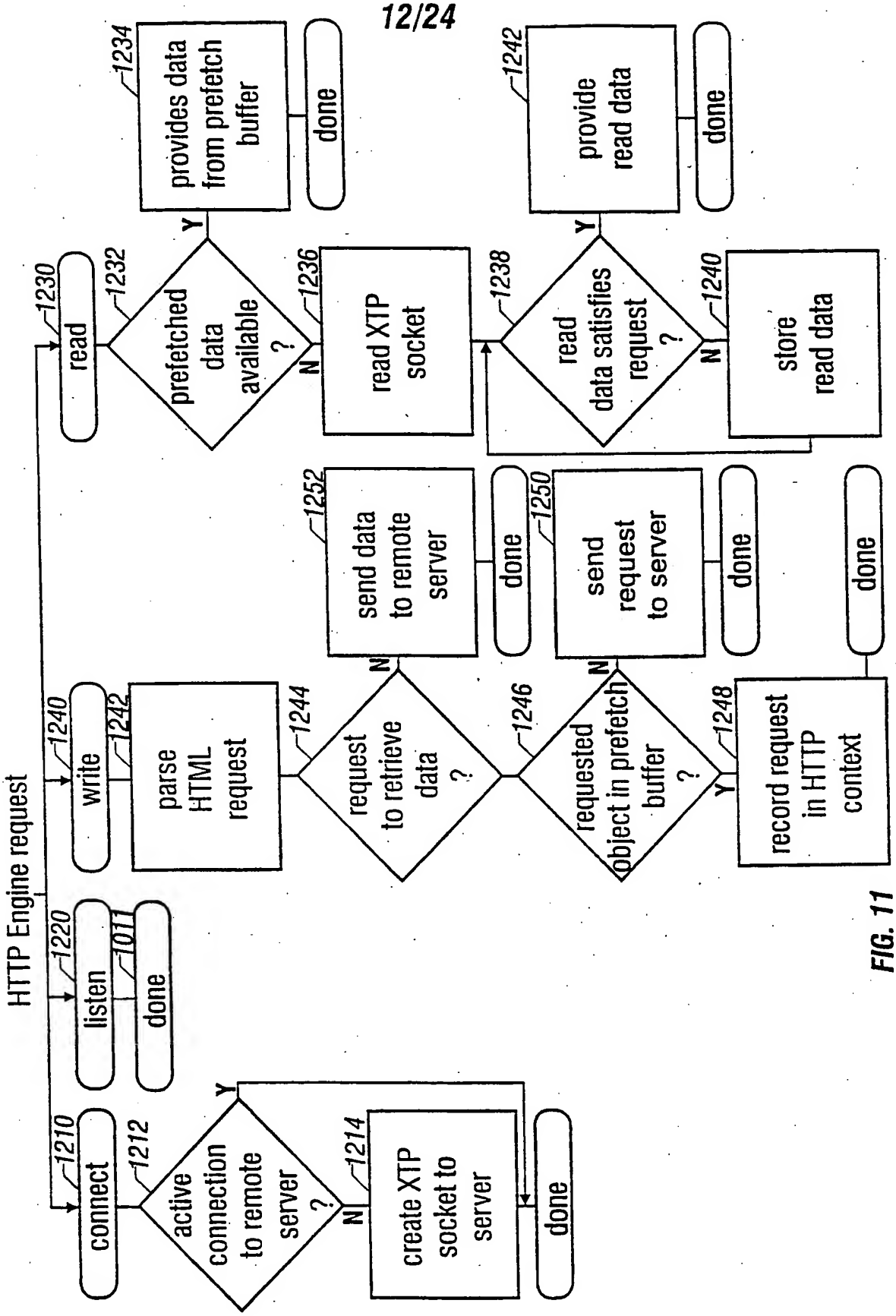

HTTP Engine request
1220
listen
1011
done
1210
connect
1212
active connection to remote server ?
Y
N
1214
create XTP socket to server
done
1240
write
1242
parse HTML request
1244
request to retrieve data ?
N
1246
requested object in prefetch buffer ?
N
Y
1248
record request in HTTP context
done
1252
send data to remote server
done
1250
send request to server
done
1230
read
1232
prefetched data available ?
Y
1234
provides data from prefetch buffer
done
N
1236
read XTP socket
1238
read data satisfies request ?
Y
1242
provide read data
done
N
1240
store read data

FIG. 11

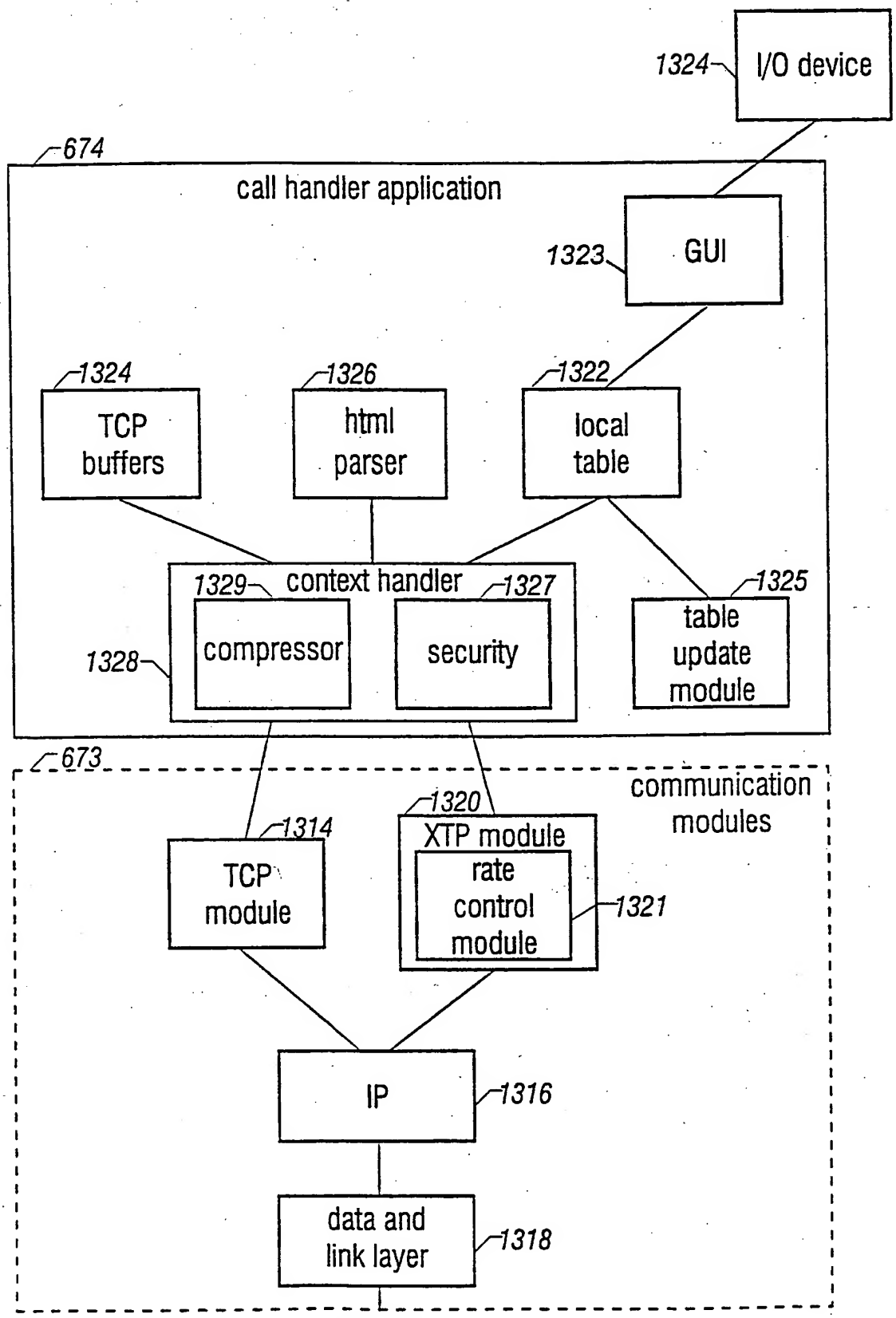

1324
I/O device
674
call handler application
1323
GUI
1324
TCP buffers
1326
html parser
1322
local table
1329
context handler
1327
1328
compressor
security
1325
table update module
673
communication modules
1320
XTP module
1314
TCP module
rate control module
1321
IP
1316
data and link layer
1318

FIG. 12

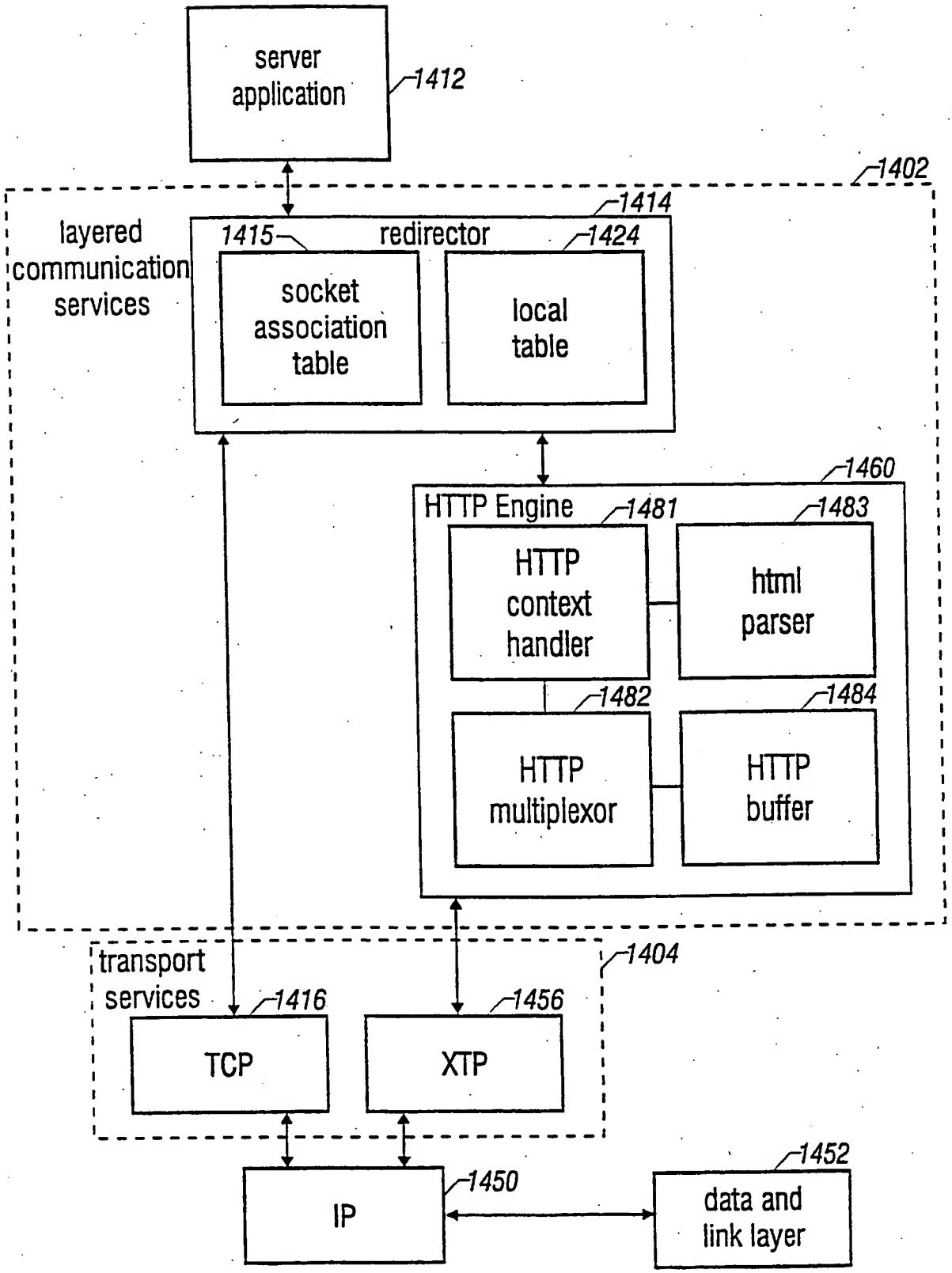

server application
1412
1402
layered communication services
1414
1415
redirector
1424
socket association table
local table
1460
HTTP Engine
1481
1483
HTTP context handler
html parser
1482
1484
HTTP multiplexor
HTTP buffer
transport services
1404
1416
1456
TCP
XTP
1452
1450
IP
data and link layer

FIG. 13

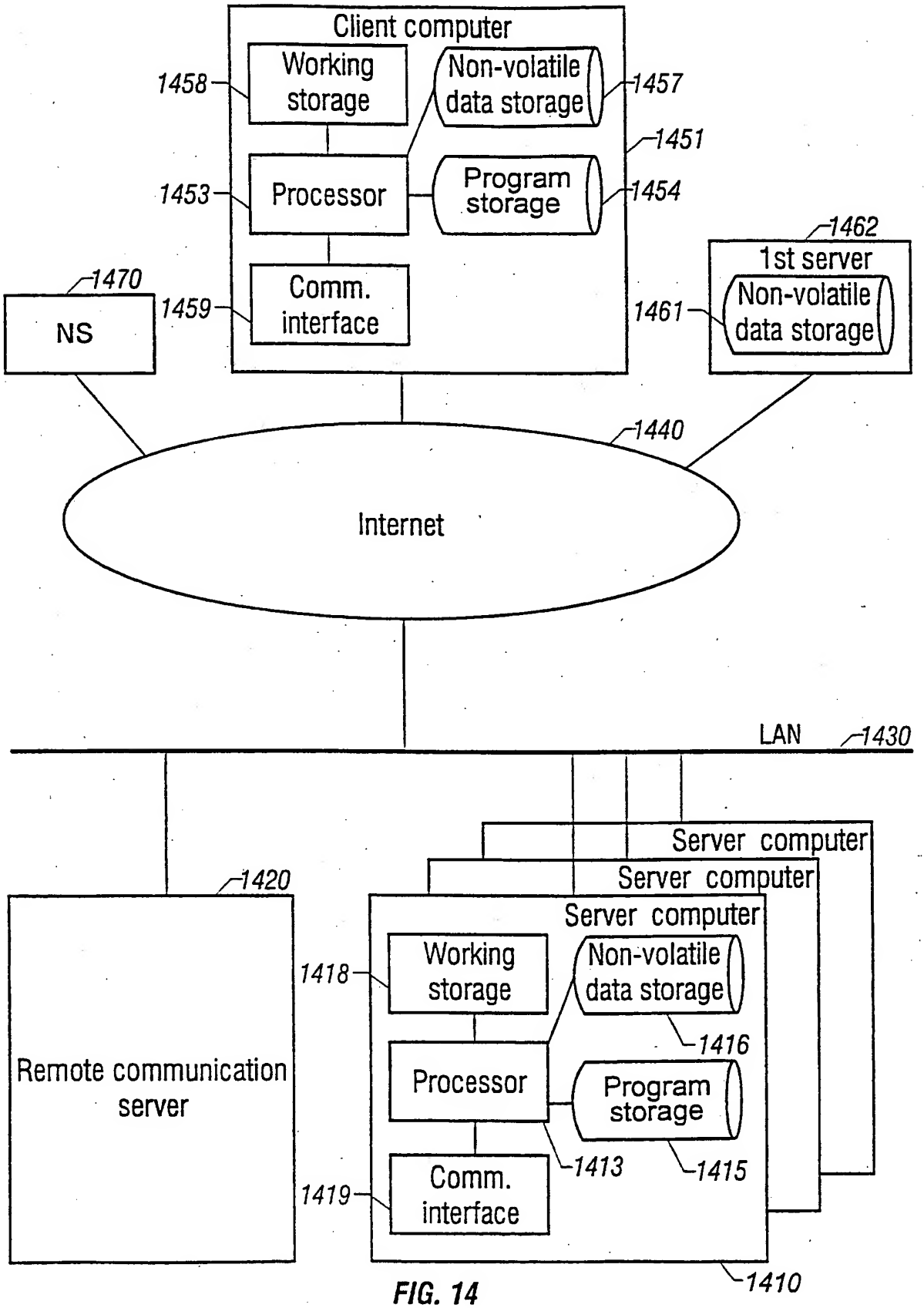

Client computer
Working storage
Non-volatile data storage
Processor
Program storage
Comm. interface
1458
1457
1451
1453
1454
1462
1st server
Non-volatile data storage
1461
1470
NS
1459
1440
Internet
LAN
1430
Server computer
Server computer
Server computer
1420
Remote communication server
1418
Working storage
Non-volatile data storage
1416
Processor
Program storage
1413
1415
1419
Comm. interface
1410

FIG. 14

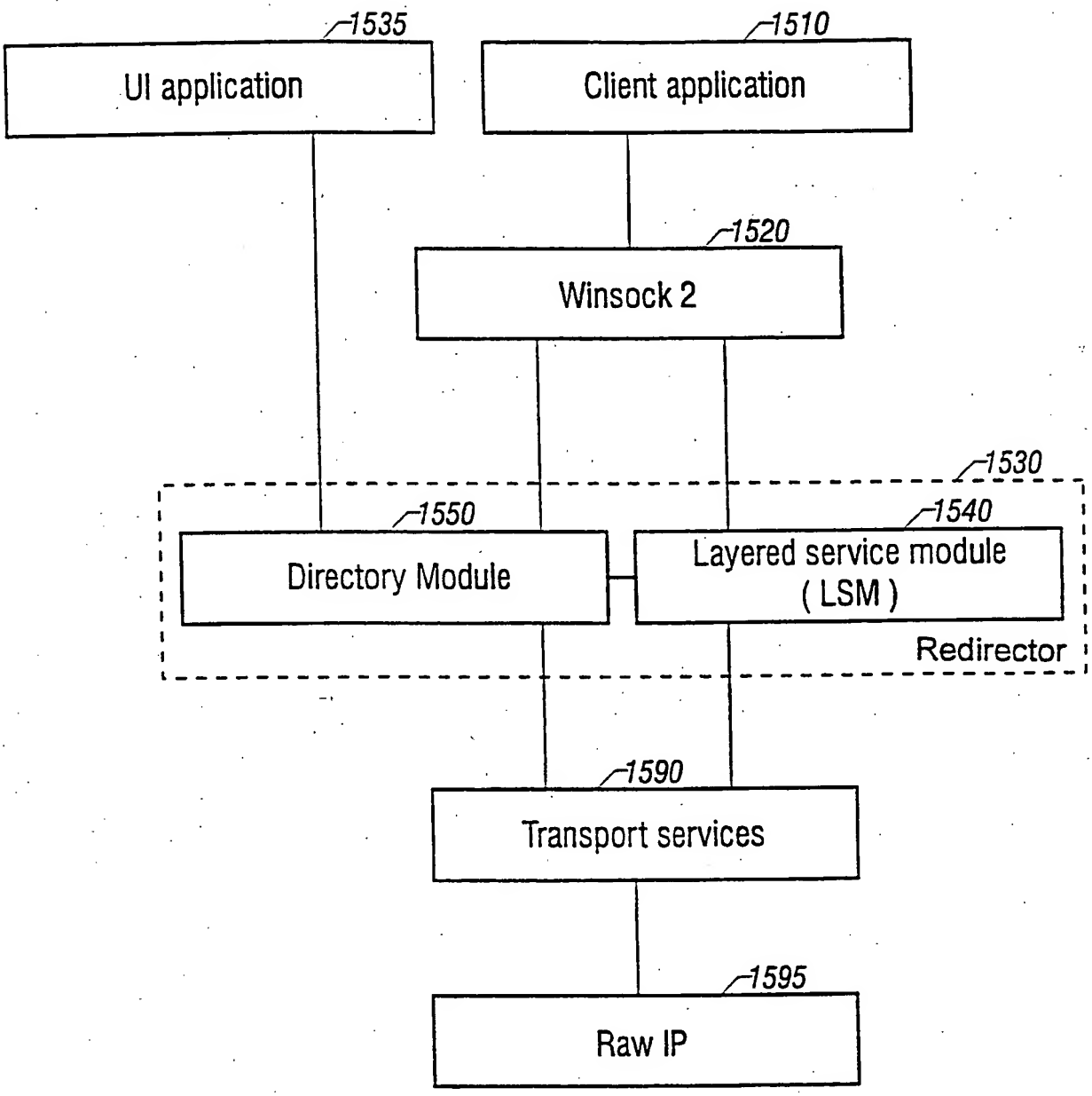

1535
UI application
1510
Client application
1520
Winsock 2
1530
1550
Directory Module
1540
Layered service module
(LSM)
Redirector
1590
Transport services
1595
Raw IP

FIG. 15

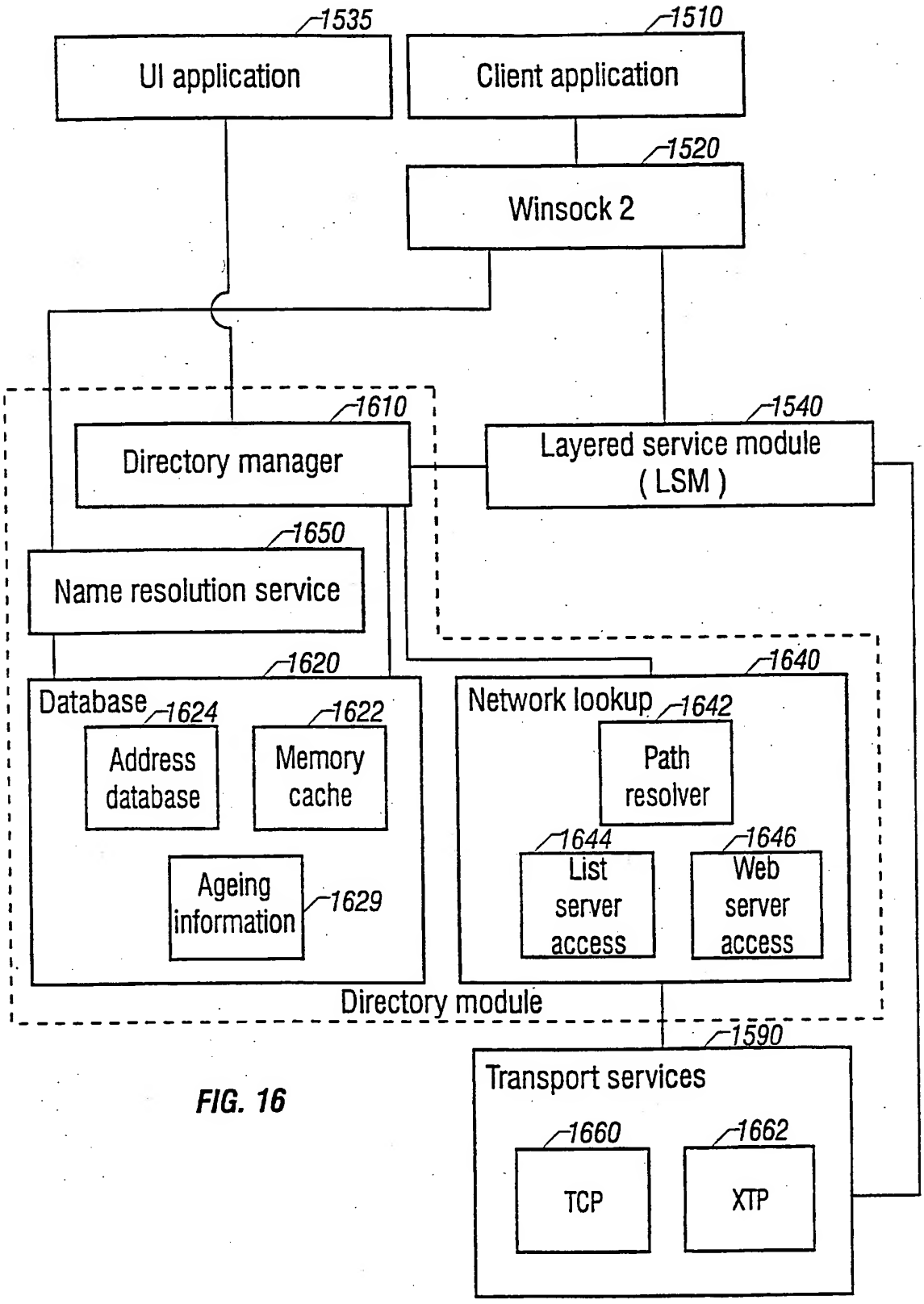

1535
UI application
1510
Client application
1520
Winsock 2
1610
Directory manager
1540
Layered service module
(LSM)
1650
Name resolution service
1620
Database
1624
Address database
1622
Memory cache
Ageing information
1629
1640
Network lookup
1642
Path resolver
1644
List server access
1646
Web server access
Directory module
1590
Transport services
1660
TCP
1662
XTP

FIG. 16

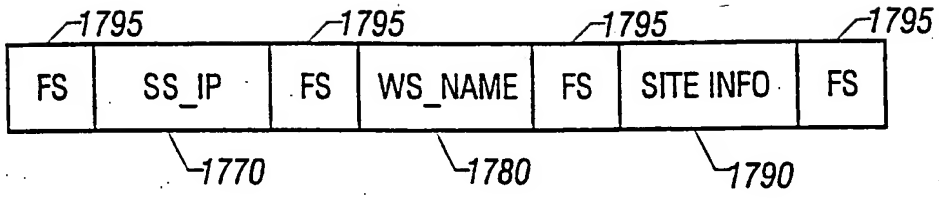


FIG. 17A

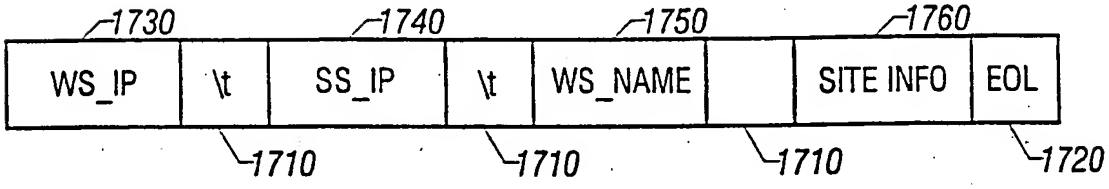


FIG. 17B

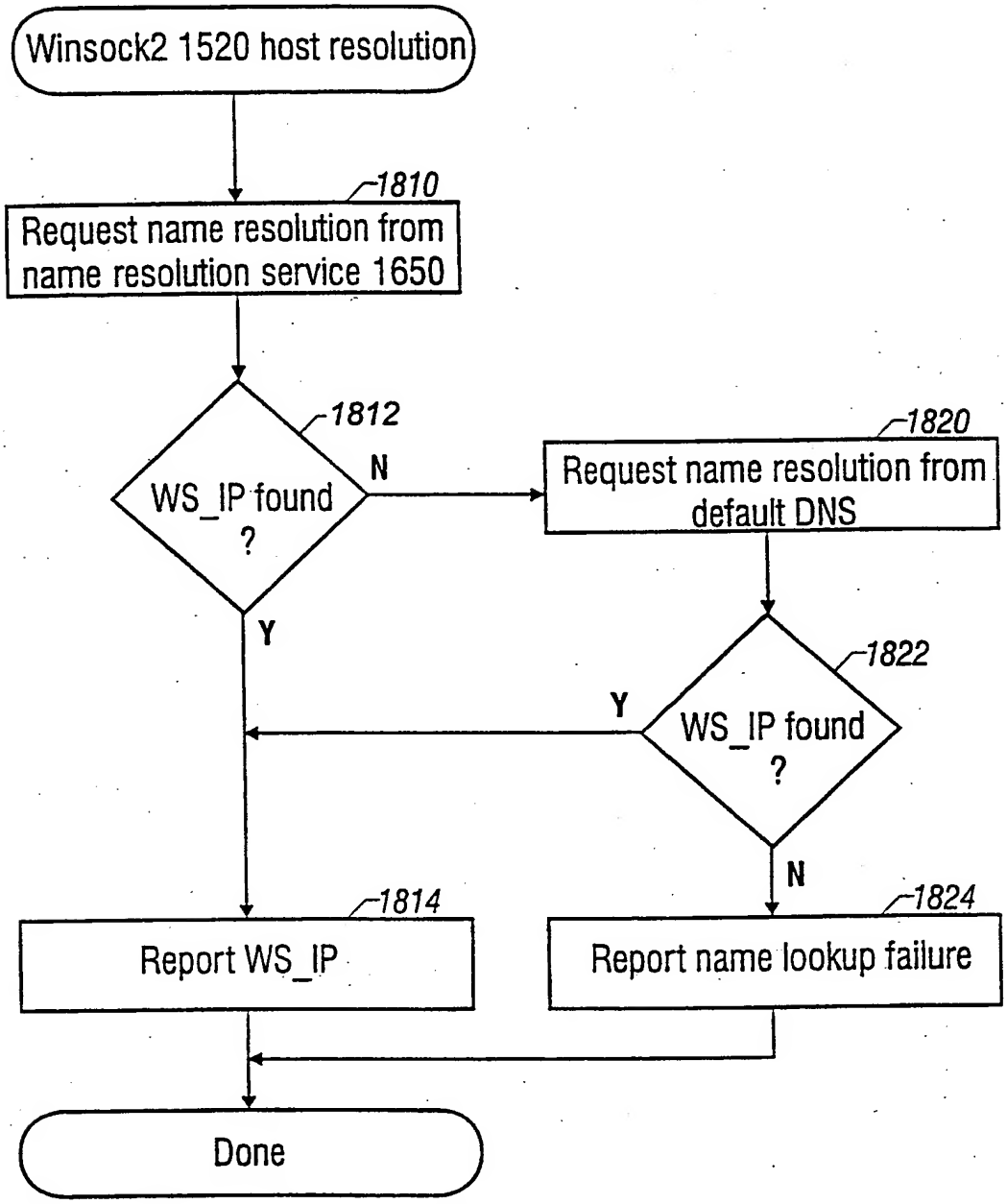

Winsock2 1520 host resolution
1810
Request name resolution from name resolution service 1650
1812
WS_IP found ?
N
Y
1820
Request name resolution from default DNS
1822
WS_IP found ?
Y
N
1814
Report WS_IP
1824
Report name lookup failure
Done

FIG. 18

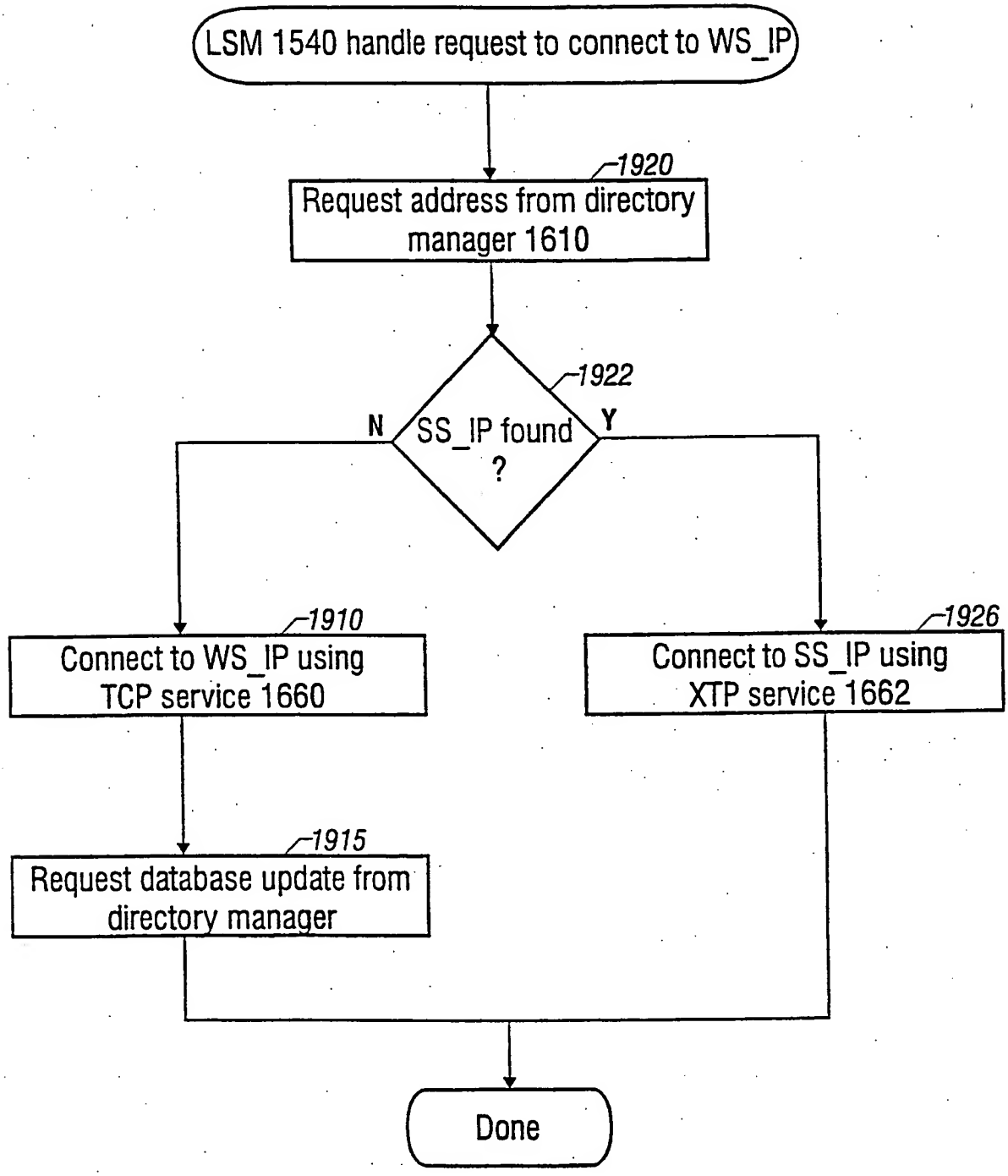

LSM 1540 handle request to connect to WS_IP
1920
Request address from directory manager 1610
1922
SS_IP found ?
N
Y
1910
Connect to WS_IP using TCP service 1660
1926
Connect to SS_IP using XTP service 1662
1915
Request database update from directory manager
Done

FIG. 19

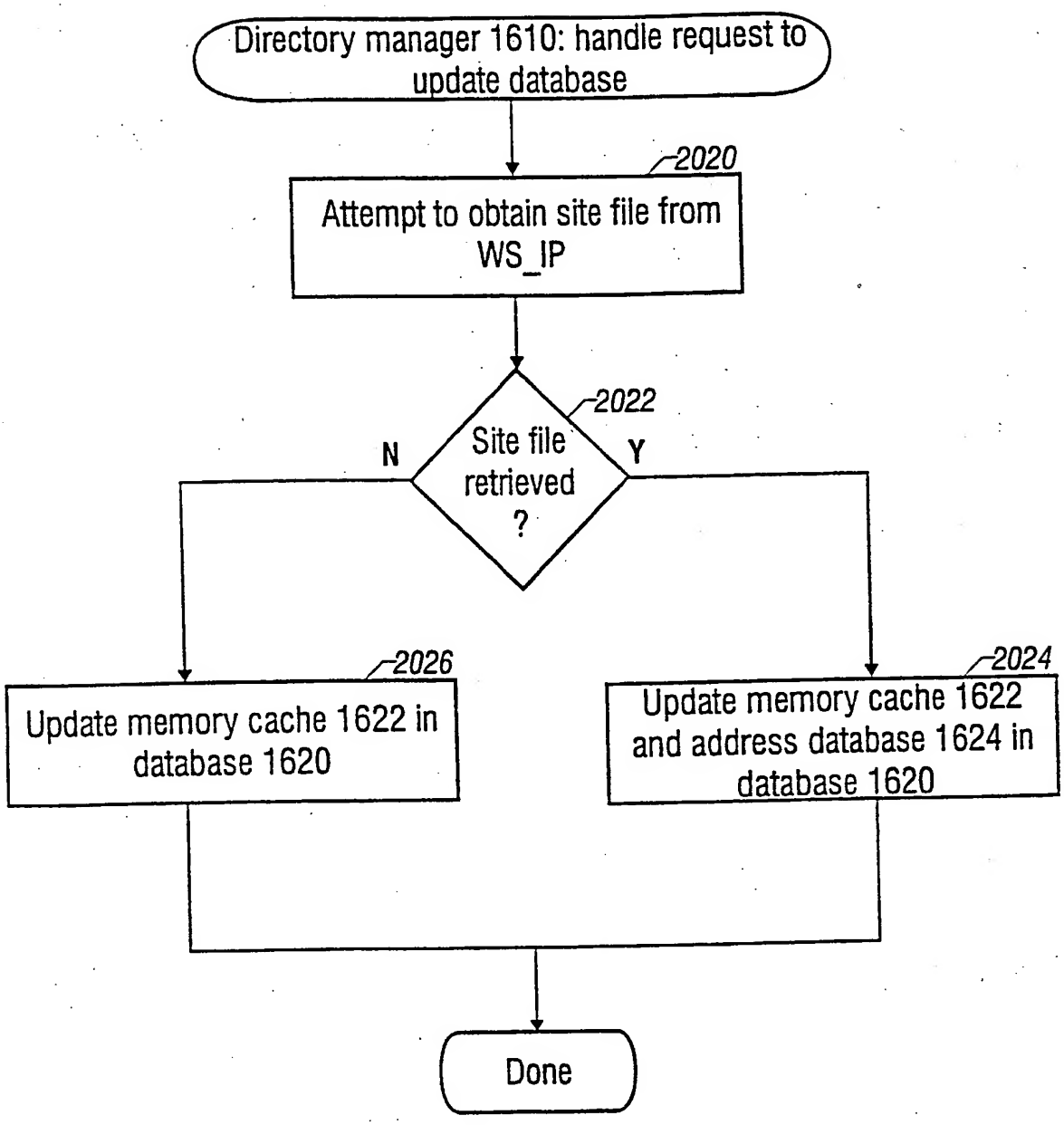

Directory manager 1610: handle request to update database
2020
Attempt to obtain site file from WS_IP
2022
Site file retrieved ?
N
Y
2026
Update memory cache 1622 in database 1620
2024
Update memory cache 1622 and address database 1624 in database 1620
Done

FIG. 20

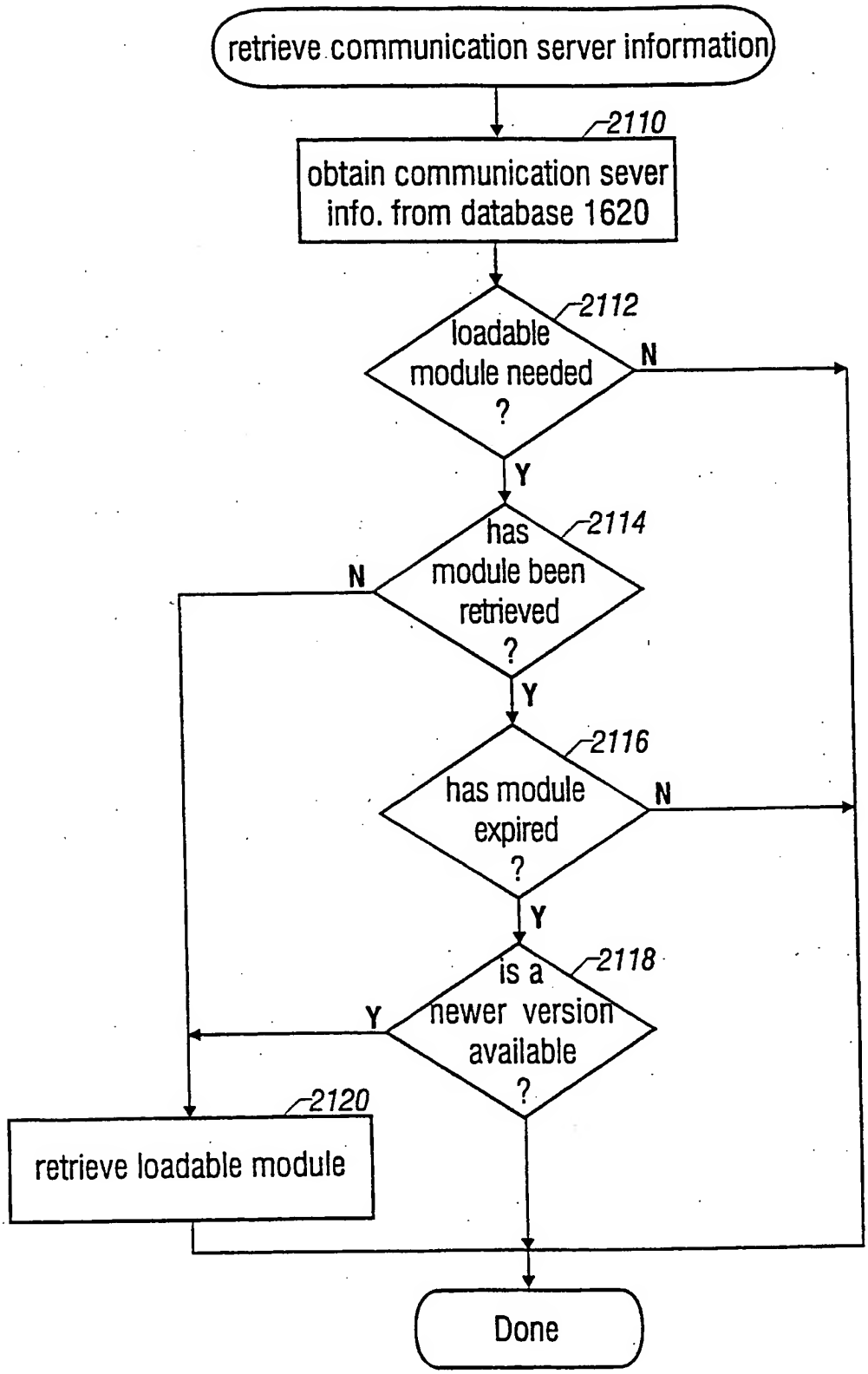

retrieve communication server information
2110
obtain communication sever info. from database 1620
2112
loadable module needed ?
N
Y
2114
has module been retrieved ?
N
Y
2116
has module expired ?
N
Y
2118
is a newer version available ?
Y
2120
retrieve loadable module
Done

FIG. 21

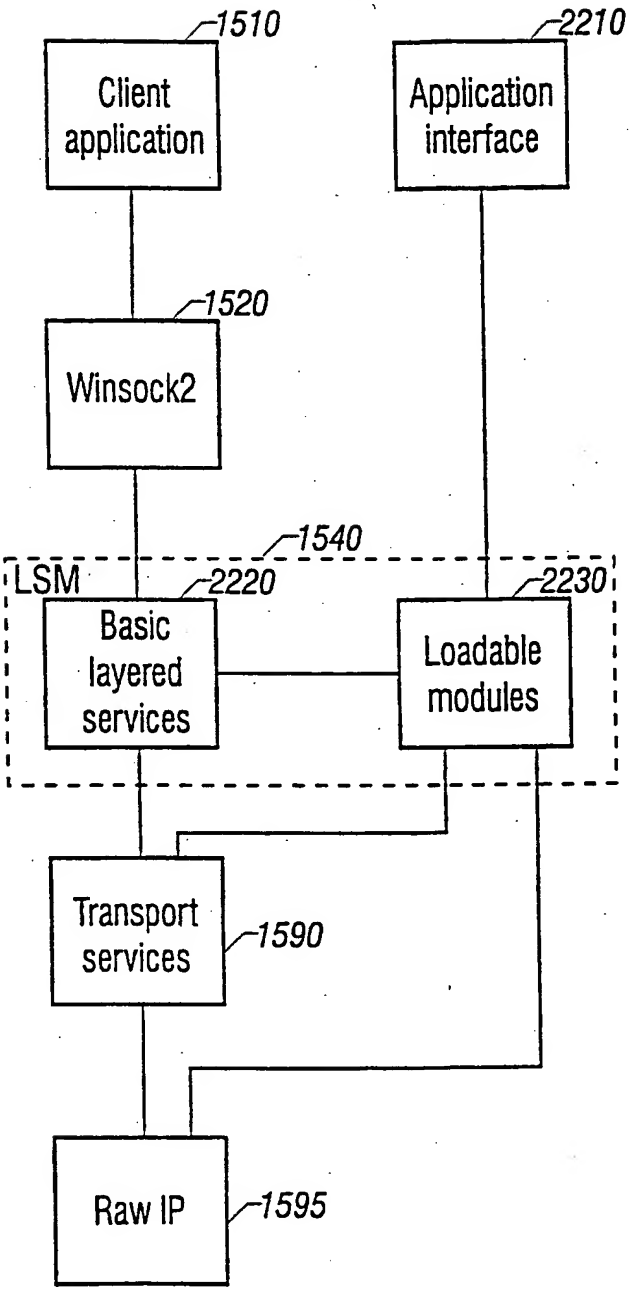

1510
Client application
2210
Application interface
1520
Winsock2
1540
LSM
2220
Basic layered services
2230
Loadable modules
Transport services
1590
Raw IP
1595

FIG. 22

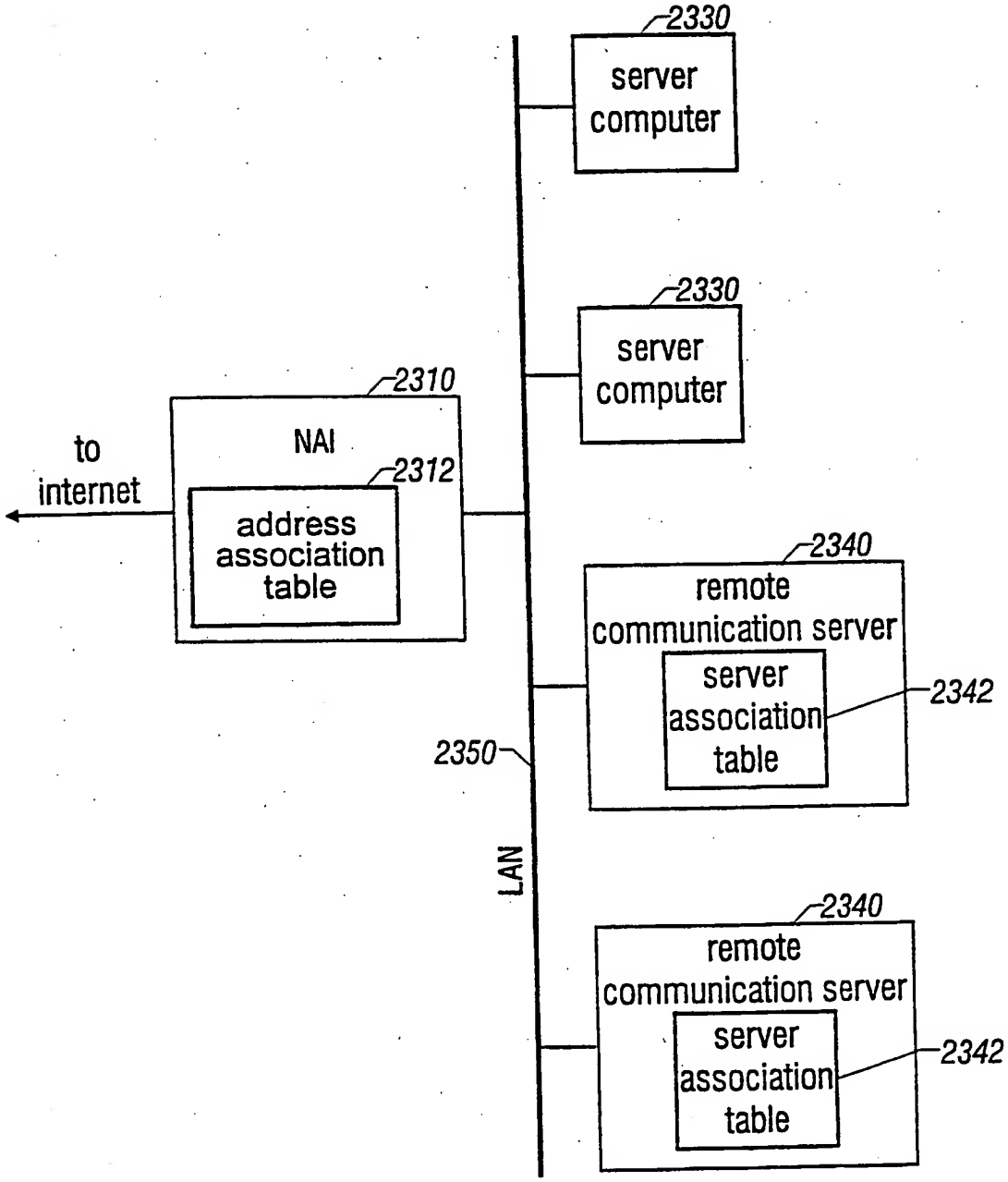

2330
server computer
2330
server computer
2310
NAI
2312
address association table
to internet
2340
remote communication server
server association table
2342
2350
LAN
2340
remote communication server
server association table
2342

FIG. 23

INTERNATIONAL SEARCH REPORT

Int itional Application No
PCT/US 98/11928

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 H04L29/06

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 751 656 A (CANON KK) 2 January 1997 see abstract see page 2, column 1, line 52 - column 2, line 26 see page 2, column 2, line 52 - line 57 see page 7, column 12, line 24 - line 55 see figure 6 --- -/--	1-5,7, 9-12,19, 22-25, 29,33, 39-41, 51,62

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
2 December 1998	09/12/1998

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Adkhis, F

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 98/11928

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 613 274 A (IBM) 31 August 1994 see abstract see page 2, line 1 - line 6 see page 3, line 2 - line 15 see page 3, line 52 - page 4, line 5; figure 1 see page 4, line 40 - line 46 -----	1-5,7,9, 19, 22-25, 39,40

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Int tional Application No
PCT/US 98/11928

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0751656 A	02-01-1997	US 5710908 A	20-01-1998
		JP 9062593 A	07-03-1997
EP 0613274 A	31-08-1994	US 5537417 A	16-07-1996
		JP 7049823 A	21-02-1995
		JP 8001622 B	10-01-1996